Conceptual Spider Diagrams

Frithjof Dau[1] and Andrew Fish[2⋆]

[1] University of Wollongong
dau@uow.edu.au
[2] University of Brighton, UK
Andrew.Fish@brighton.ac.uk

Abstract. Conceptual Graphs are a common knowledge representation system which are used in conjunction with an explicit type hierarchy of the domain. However, this means the interpretation of information expressed in conceptual graphs requires the combined use of information from different sources, which is not always an easy cognitive task. Though it is possible to explicitly represent the type hierarchy with Conceptual Graphs with Cuts, this less natural expression of the type hierarchy information is not as easy to interpret and soon takes up a lot of space. Now, one of the main advantages of Euler diagram-based notations like Spider diagrams is the natural diagrammatic representation of hierarchies. However, Spider diagrams lack facilities such as the ability to represent general relationships between objects which is necessary for knowledge representation tasks. We bring together the most pertinent features of both of these notations, creating a new hybrid notation called Conceptual Spider Diagrams. We provide formal syntax and semantics of this new notation, together with examples demonstrating its capabilities.

1 Introduction

Contemporary knowledge processing systems that include inferential abilities are typically based on some variant of formal logic where the information is internally stored in a particular format according to the sentences of the logic used. Such formal logics and their reasoning mechanisms have been thoroughly investigated and form a solid background for knowledge processing systems. However, the representation of knowledge as formulae has drawbacks if they are to be used for communication: in particular, they can be hard to comprehend by readers who are untrained in mathematics.

In contrast with the usual formal logics, human reasoning is often multimodal, involving information obtained from sentences, diagrams, sound, nuance or moving pictures for instance. The research field of *diagrammatic reasoning* investigates all forms of human reasoning and argumentation wherever diagrams are involved. Diagrams are often deemed to be easier to comprehend than symbolic notations [20, 23, 25], especially when they make good use of spatial relationships which are not utilized in symbolic notations; in particular it has been argued that they are useful for knowledge representation systems [8, 19].

⋆ funded by UK EPSRC grant EP/E011160: Visualisation with Euler Diagrams.

There are two major families of mathematically well-elaborated diagrammatic reasoning systems, called conceptual graphs (CGs) and spider/constraint diagrams (SDs/CDs), which both have their roots in the works of Charles Sanders Peirce. Firstly, Sowa's CGs are based on Peirce's Existential Graphs. Various fragments of Sowa's CGs have been developed in a precise manner based on graph theory (see [2] for an overview), and in these it is possible to represent and carry out reasoning with relations of arbitrary arity. Secondly, SDs and CDs are based on Euler diagrams which are closely related to Euler circles and Venn-Peirce diagrams, and provide explicit means to easily represent certain relationships between sets (such as subset and disjointness). Various systems of SDs and CDs have been developed and formalized using algebraic means.

Now, although SDs and CDs are effective at expressing and reasoning with the relationships between sets, since the spatial relationships which encode them are well-matched [10], their usefulness in representing and reasoning with arbitrary relations is not so clear. On the other hand, CGs use a convenient representation of relations, which were in fact designed for this purpose, but their only means to express relationships between sets is to employ an underlying type-hierarchy. This has the drawbacks that: it only allows the expression of subset-superset relationships; it is a different representation of information, formally separated from the conceptual graphs and is often not even displayed with the CGs. This paper provides a step towards unifying these two diagrammatic systems, drawing on the specific advantages of each system and overcoming some of their disadvantages. We will use the underlying notation of SDs in order to make the type-hierarchy[1] of CGs both more explicit and more expressive, and we will augment this notation with relations of arbitrary arity as is done in CGs. The resulting notation will be called *conceptual spider diagrams*.

Due to limited space, we will assume that the reader is familiar with CGs; see [2, 26] for details. In section 2 we provide an exposition of the Euler diagram variants with a particular emphasis on SDs. We compare features of the SD and CG notations in section 3, identifying good and bad properties, as they relate to knowledge representation. An introduction to our hybrid notation via a collection of simple examples is provided in section 4, where the usefulness of the combined features becomes clear. A formalisation of the syntax and semantics of Conceptual Spider Diagrams is given in section 5. This opens up many interesting avenues of future research, and some of these are mentioned, together with our concluding remarks, in section 6.

2 Euler Diagram based systems

In seminal work [24], Shin produced a sound and complete formal diagrammatic reasoning system based on an extension of Venn diagrams, and logical reasoning systems based on Euler diagrams are now commonplace [4, 11, 15, 29, 31].

[1] To be more precise: this paper tackles the type-hierarchy of the *concepts*: relations are not adressed, as a convenient way for diagrammatically depicting subset-superset relationships between relations goes beyond the abilities of SDs.

Spider diagrams (SDs) and constraint diagrams (CDs) are diagrammatic reasoning systems based on Euler diagrams which are applied in a multitude of areas including: file-information systems, library systems, statistical data representation and for logical software specification and reasoning systems. In this section we give a brief review of this family of diagrams, starting from Euler diagrams, passing through Venn-Peirce diagrams and arriving at SDs and CDs.

Euler first introduced Euler circles [3] in which sets are depicted by circles, and the spatial relationships between the circles mimic the set-theoretic relationships. The modern variant of these which we describe are called Euler diagrams: they use simple closed curves in the plane to represents sets. Two curves do not overlap (or more precisely, the interiors of the discs bounded by the curves are disjoint) if and only if the corresponding sets are disjoint, and if one circle is contained in another circle (or, more precisely, the interior of one is contained in the interior of the other) then we have a subset relationship between the corresponding sets. An example of an Euler diagram is shown on the left of Fig. 1. The interpretation is that there are three sets A, B and C such that B is a subset of A and that B and C are disjoint; note that no information about the relationship between A and C is provided.

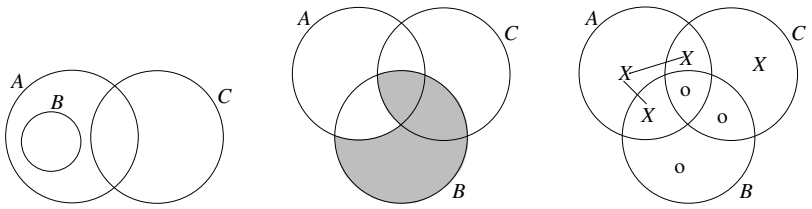
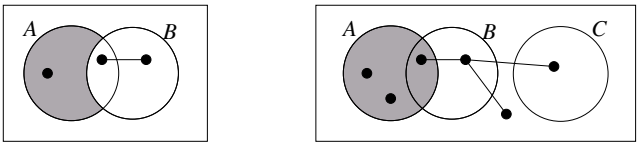


Fig. 1. An Euler diagram, a Venn diagram, and a Venn-Peirce diagram

There are collections of set intersections that cannot be represented using simple closed curves due to natural topological constraints [21], and so shading in a region is used in order to express emptiness. Then, even restricting to Venn diagrams [32], which are a subclass of Euler diagrams in which all possible set intersections are represented, one can depict any collection of set intersections. The middle diagram in Fig. 1 is a Venn diagram with exactly the same meaning as the Euler diagram. Using shading within the Euler diagram system allows a greater flexibility of representation than just using Venn diagrams with shading, but it is still not possible to express that sets are non-empty.

Peirce augmented Venn diagrams by X-sequences, which denote an element of the set corresponding to the region in which the sequence is placed. In his system, the shading of Venn diagrams is replaced by placing an 'o' in the region. The righthand diagram of Fig. 1 is a Venn-Peirce diagram which again expresses that $B \subseteq A$ and $B \cap C = \emptyset$ holds by the use of the o's; it also expresses that there is an element in $A \backslash (A \cap B \cap C)$, depicted by the X-sequence with three Xs, and that there is an element in $C \backslash (A \cup B)$.

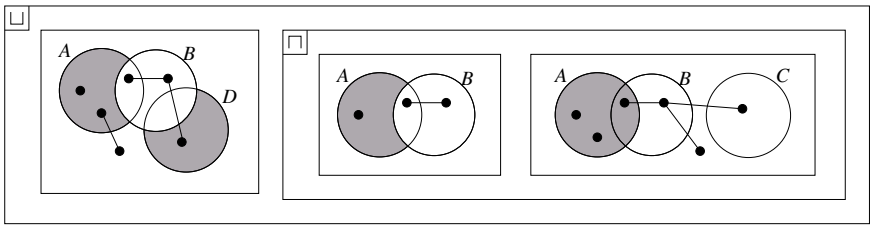
Since the first paper on SDs [9] appeared, several elaborations and variations of the ideas have been published [33]. Two examples of SDs are shown below.



By convention a bounding rectangle is used to depict the UNIVERSE OF DISCOURSE, which is the ground set of the respective models in which the diagrams are evaluated; the letter U is used to denote this universe. The lefthand diagram above has two CONTOURS – the curves labeled A and B – representing the sets A and B, and four ZONES representing all possible set intersections involving A and B. The diagram contains two spiders, and the shaded zone representing $A - B$ is the HABITAT of the first spider, whilst the habitat of the second spider is the REGION representing B which is composed of two zones representing $A \cap B$ and $B - A$. The interpretation is that there are two sets A and B, the set $A - B$ contains exactly one element, and the set B contains at least one element.

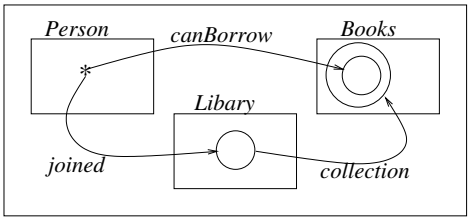
So, there are two important differences between the semantics of Venn-Peirce diagrams and those of SDs: for SDs different spiders necessarily denote different objects, whilst the objects represented by different X-sequences in Venn-Peirce diagrams are not necessarily distinct; the shading in SDs means that a region does not contain *more* elements than the elements represented by the spiders touching that region, whilst for Venn-Peirce diagrams the 'o' indicates emptiness independently of any impinging X's. Now, in the righthand diagram there is another contour representing a set C and it does not overlap with the contours representing A and B and so C and $A \cup B$ are disjoint. Moreover, there are three distinct elements u, v and w (represented by the spiders) such that $u, v \in A$ and $w \in U - (A - B)$. Due to the shading the set $A - B$ contains exactly u and v, and $A \cap B$ either contains no elements or exactly w.

The above diagrams are called UNARY SDs, but spider diagrams can also be propositionally combined using the logical operators $\sqcap$ ('and') and $\sqcup$ ('or'): an example which uses these conjunctors is shown below.



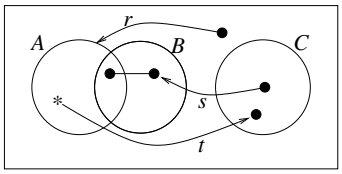
Similar to CGs, an important facet of SDs is that they are not only diagrammatic *representation* languages, but that they also facilitate diagrammatic *reasoning*. The system of SDs [16] is equipped with a sound and complete calculus. Automatic theorem provers for Euler diagrams and SDs have been developed and implemented [7, 28], with the goal of aiding the user's understanding of proofs. A tableaux system for SDs has also been implemented by Patrascoiu [22].

Finally, CDs were first proposed by Kent [18] as a notation for visualizing object-oriented invariants in software modelling and they can be used to express operation pre-conditions and post-conditions for modelling purposes [14]. They extend SDs, allowing the explicit representation of universal quantification via an extra type of spiders, called *universal* spiders which are indicated by asterisks, and arrows which provide information about binary relations. An example of a CD is show below, where the core information expressed is that "every person can only borrow books that are in the collections of libraries they have joined".



Since Kent's informal idea was proposed, several papers have provided various levels of mathematical formalism for Kent's vision. Problems such as the ambiguities in ordering of quantifiers have been solved by augmenting with an explicit reading tree in [5] where formal syntax and semantics for constraint diagrams are provided. Although some rules have been developed [4], a sound and complete calculus has not yet been developed for this system. However, for certain restricted fragments of the system, sound and complete calculi have been developed (e.g. in [27] no universal quantifiers were allowed as well as severe restrictions on the allowable relations and in [30] the restriction to order all of the universal quantifiers after all of the existential ones was imposed).

We elaborate briefly on the definition of relations between spiders in these systems. Arrows place restrictions on the relations determined by their labels. In particular if an arrow labelled R has source a spider s and target a spider t then the semantic phrase "$S.R = T$" is associated with this arrow (where S is the object represented by s and similarly for T, and $S.R := \{x \mid (S, x) \in R\}$).



Using the semantics in [5], but adopting one of the conventions of [30] of reading the universal spiders after the existential ones (removing the need for an explicit depiction of a reading tree), a reading of the diagram is given by:

$$\overbrace{C \cap (A \cup B) = \emptyset}^{\text{Euler notation}} \wedge \overbrace{\exists u \in U - (A \cup B \cup C).\, \exists b \in B.\, \exists c_1, c_2 \in C :}^{\text{objects denoted by (existential) spiders}}$$
$$(c_1 \neq c_2 \wedge \underbrace{u.R = A \wedge c_1 S = b \wedge \forall x \in A - B : x.T = \{c_2\}}_{\text{reading the arrows}}).$$

3 Desirable features of a hybrid notation

We investigate the advantages and disadvantages of some features of CGs and SDs that are useful for knowledge representation reasoning system with the aim of building a hybrid notation using the most pertinent features.

CGs are based on an underlying type hierarchy, which is usually a partially ordered set that indicates the subtype/supertype relationships between the types. This type hierarchy is handled as some sort of background information, not explicitly appearing in the actual CGs, but it is used for reasoning purposes.[2] This separation of information can make reasoning more difficult for humans. On the other hand, SDs provide an effective method for expressing the relationships between sets, since they are based on Euler diagrams. Thus if one combined the Euler diagrams features with a CG then one could make the type hierarchy of CGs explicit thereby making it visible to the user and aiding them in reasoning tasks; since SDs depict relationships between sets in a very iconic manner, the user will not only read off only the information which was needed to construct the type hierarchy, but other information which can be deduced from the type hierarchy can often easily been read of the SDs as well. This advantage of icons is described by Peirce in [12], 2.279, where he writes that "a great distinguishing property of the icon is that by the direct observation of it other truths concerning its object can be discovered than those which suffice to determine its construction." In the modern research field of diagrammatic reasoning, Shimojima coined the term *free ride* for this property of iconic representation [23].

Another advantage of bringing the type hierarchy to the foreground, using SDs is that we extend the expressiveness of the type hierarchies. Whilst the existing type hierarchies used only express subtype/supertype relationships between the types, taxonomies often contain disjointness information as well, and since disjointness constraints as well as subset relations can be naturally expressed with SDs, we can overcome this limitation.

When building a hybrid system, it pays to consider the semantic options carefully. The choice of shading over o's increases the expressiveness by allowing the expression of upper bounds on the cardinality of sets. The choice of using spiders over X-sequences is an interesting choice, with the spider semantics adhering closely to diagrammatic design principles (distinct spiders meaning distinct objects) whilst the use of X-sequences closely matches the usual symbolic logic rules. We will actually investigate both options for the semantics of spiders (hereafter we will refer to the SD-semantics or the VP-semantics of spiders instead of spiders versus X-sequences).

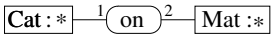
When considering relations between objects, we notice that in CDs, the semantics for arrows is significantly different to the semantics for binary relations in FOL or CGs. For example, if we have two spiders standing for objects u and v, an arrow between these two spiders does not mean that the corresponding

[2] If projections are chosen for reasoning, we have appropriate conditions for the projections which reflect the type hierarchy. If transformation rules are used instead, we usually have type-generalization rules among these transformation rules.

objects stand in relation R (i.e. uRv). Instead, we have the stronger condition $u.R = \{v\}$ (i.e. uRv and there is no object $w \neq v$ with uRw). Moreover, constraint diagrams do not allow relations with an arity > 2 (although this could be addressed by using multi-sourced or targetted arrows). Thus we choose to augment SDs with a different means to express relations between objects, adapted from the handling of relations in CGs.

4 Hybrid System examples

Before we come to the formalization, we will exemplify our approach with the following well known toy example from the CG-community:



The meaning of this CG is 'there is a cat and there is a mat, and the cat is on the mat'; according to the usual semantics of CGs, it is not guaranteed that the cat and the mat are different objects. We assume that we have an underlying type hierarchy, with types $\top$, Animal, Cat, Thing, Mat, and Rug. The type-hierarchy is a partially ordered set, indicating only the supertype/subtype relations, and is shown on the left of Fig. 2. On the right of the figure an extension of this type-hierarchy is depicted, where disjointness-constraints are added.

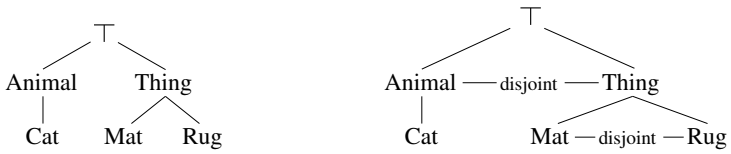


Fig. 2. The underlying type hierarchy: without, and then with disjointness constraint

Now, if one wished to *explicitly* display this type hierarchy in CGs, we need the facility to express supertype-subtype relations and disjointness constraints. This can be done with CGs with cuts, as shown in Figure 3, but obviously this CG suffers from readability problems. Alternatively, we can depict the type hierarchy with an Euler Diagram, as shown in Fig. 4. The $\top$-concept corresponds to the universe of discourse, so there is no need to have a contour for it (although one could think of the rectangular bounding box for the diagram as signifying this concept). This diagram is significantly more readable than the CG of Fig. 3, and this may be partly due to free rides: some information which is only implicitly given in the CG in Fig. 3 or from the diagram in Fig. 2 (e.g. that that types Cat and Mat are disjoint) can immediately be read off from the Euler diagram.

We now demonstrate our proposed hybrid notation for the cat-on-mat example. In the lefthand diagram of Fig. 5, the Euler diagram of Fig. 4 is augmented with two spiders, denoting a cat and a mat, and the relationship 'on' between

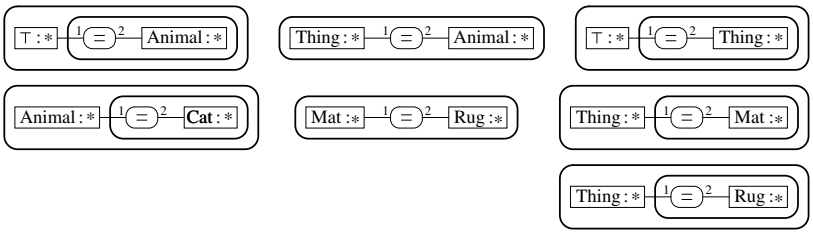


Fig. 3. The underlying type hierarchy with disjointness constraints as a CG with Cuts

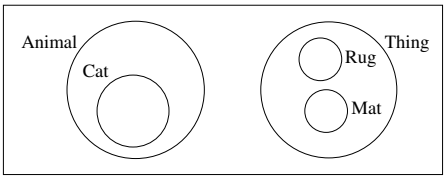


Fig. 4. The type hierarchy with disjointness constraints as an Euler Diagram

them. This diagram contains the complete type hierarchy plus the information that there is a cat on a mat. In the righthand diagram, we show only the fragment of the type hierarchy that is necessary to express the 'there is a cat on a mat' statement (in real-life scenarios, it will be a choice of the knowledge engineer which part of the type-hierarchy is actually displayed). This diagram is still strictly more expressive than the initial CG of this section: since the contours for cat and for mat do not overlap, this diagram also contains the information that *the cat and the mat are different*. We note that in contrast to the spiders, the location of the relation-symbol 'on' is not semantically important.

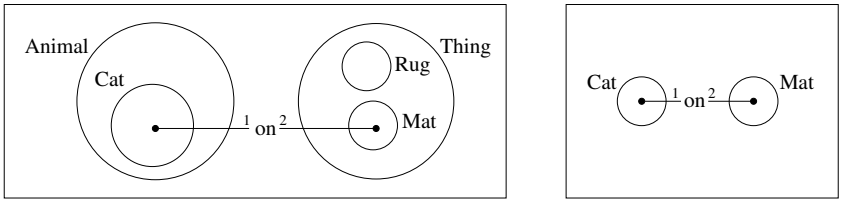


Fig. 5. Two hybrid diagrams for 'a cat is on a mat'

The next diagram shown in Fig. 6 is more sophisticated, and since it involves distinct spiders touching the same region, we consider both the SD-semantics and the VP-semantics. As well as the hierarchy information, it expresses that Yoyo is a cat, there is an unnamed cat (which is different from Yoyo in the SD-semantics, but possibly the same as Yoyo in the VP-semantics), and there exist no other cat (this holds for both semantics given the interpretation of shading), Yoyo is on a mat, and the unnamed cat is either on a rug or a mat (which is different from Yoyo's mat in the SD-semantics, but possibly the same in the VP-semantics).

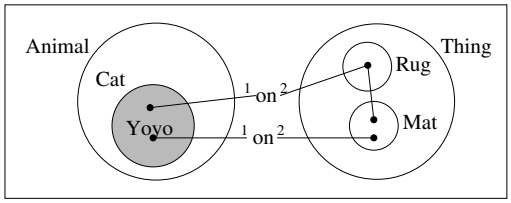
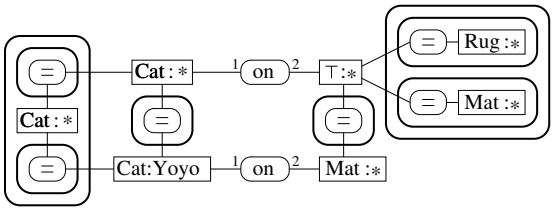


Fig. 6. A more sophisticated hybrid diagram

If we express this information by means of a CG with cuts, even without expressing the type-hierarchy information, then the outcome is significantly less readable. The diagram below shows this, using the SD-semantics; the corresponding CG for the VP-semantics can be obtained by removing the two vertical, negated identity edges in the middle of the CG making the diagram only slightly more comprehensible.



5 Formalisation

When considering the formalization of diagrammatic logic systems, it is essential to realize that there are two levels of notation. Certain graphical features of the diagram may be varied without changing the meaning of the diagram. For example, the shape of the contours in Euler diagrams or the shape of cuts in CGs with cuts is not semantically important; nor does the positioning of the boxes of a CG matter. Thus we have to deal with an *abstract syntax* which prescinds the meaningful information from such graphical properties, and a *concrete syntax*, which corresponds to the actual drawings of diagrams. This vital distinction has been discussed in the CG-community [1] as well as in the SD- and CD-community [13]; both papers argue that it is essential to elaborate the abstract syntax in a formal manner, but differ in their opinions concerning the concrete syntax. In [1] diagrammatic systems like existential or conceptual graphs are considered, where every diagram at the abstract level has a graphical representation (i.e. a corresponding concrete diagram), and argues that a mathematical formalization of the concrete syntax is *not* needed. However, for Euler diagrams the situation is different, since not all collections of sets intersections can be graphically represented without introducing some extra set intersections (or zones). That is, there are abstract Euler diagrams which are not *drawable* using simple closed curves, without introducing shaded zones, and although this introduction is not particularly problematic in logical reasoning systems it might be very undesirable

in information display systems. So unsurprisingly, [13] argues that the concrete syntax as well as the relationship between the abstract and concrete syntax has to be *mathematically* formalized as well.

The methodology usually employed is to build Euler diagram based reasoning systems at the abstract level and then generate concrete diagrams (under various sets of wellformedness conditions) when it is possible to do so. On the positive side, from a human interaction point of view, the automatic generation of concrete Euler diagrams from abstract Euler diagrams for a strict set of wellformedness conditions is possible [6]. When extending to spider and constraint diagrams no extra substantial representational difficulties occur. The conceptual spider diagrams that we introduce are also extensions of spider diagrams and so they have to cope with these same difficulties, but the augmentation to express relations does not add any complexity to this issue either. For this reason, we introduce only the abstract syntax of conceptual spider diagrams (although there is no real problem in also defining the concrete syntax formalization).

5.1 Conceptual Spider Diagrams

Definition 1 (Alphabet). *An* ALPHABET *is a triple* $\mathcal{A} := (\mathcal{O}, \mathcal{C}, \mathcal{P})$ *of disjoint sets* $\mathcal{O}$, $\mathcal{C}$, $\mathcal{P}$ *of* OBJECT NAMES, CONCEPT NAMES *and* PREDICATE NAMES, *respectively. To each predicate name* P, *we assign its* ARITY $ar(P)$. *Let* $*$ *be another sign, the* GENERIC MARKER. *We set* $\mathcal{O}^* := \mathcal{O} \dot\cup \{*\}$.

In the examples provided earlier, object names or the generic marker $*$ were used as the labels of spiders, concept names were the labels of contours, and predicate names were the labels of predicate edges. An abstract zone represents a particular set intersection, and as such it will be defined to be a pair of disjoint, finite sets (a, b), where the set of contour labels a are those of the containing sets and the set of contour labels b are the excluding sets (i.e. the rest of the contour labels). In a unary diagram specifying the containing set a is sufficient (since a and b form a partition of the set of concept names $\mathcal{C}$), but when considering non-unary diagrams, or when reasoning with diagrams, the contour label sets may vary and so the pair of sets is required. Let $\mathcal{Z}$ and $\mathcal{R} := \mathfrak{P}(\mathcal{Z})$ denote the sets of all zones and regions respectively, where $\mathfrak{P}$ denotes the power set.

Definition 2 (Conceptual Spider Diagram). *An* UNITARY CONCEPTUAL SPIDER DIAGRAM *(abbreviated* UNITARY CSD*), d, over an alphabet* $\mathcal{A}$ *is a tuple* $(L, Z, Z^*, S, \eta, E, \nu, \kappa)$ *whose components are as follows:*

1. $L := L(d) \subseteq \mathcal{C}$ *is a finite set of concept names.*
2. $Z := Z(d) := \{(a, L - a) : a \subseteq L\}$ *(with* $Z(d) \subseteq \mathcal{Z}$*) is a set of zones s.t.*

$$(i) \quad \forall l \in L\ \exists (a, b) \in Z : l \in a \qquad \text{and} \qquad (ii) \quad (\emptyset, L(d)) \in Z\ .$$

   We define $R(d) = \mathfrak{P}(Z) - \{\emptyset\}$ *to be the set of regions in d.*
3. $Z^* := Z^*(d) \subseteq Z$ *is a set of* SHADED ZONES. *We define* $R^*(d) = \mathfrak{P}(Z^*) - \{\emptyset\}$ *to be the set of* SHADED REGIONS *in d.*

4. $S := S(d)$ *is a finite set of* SPIDERS *with* $S(d) \cap (\mathcal{C} \cup \mathcal{Z} \cup \mathcal{R}) = \emptyset$.
5. *A function,* $\eta := \eta_d : S(d) \to R(d)$ *which returns the habitat of each spider.*
6. $E := E(d)$ *is a finite set of* PREDICATE EDGES *with* $E(d) \cap (\mathcal{C} \cup \mathcal{Z} \cup \mathcal{R}) = \emptyset$.
7. *A function,* $\nu := \nu_d : E(d) \to \bigcup_{n \in \mathbb{N}} (S(d))^n$ *which returns for each edge an n-tuple* ($n \in \mathbb{N}$) *of spiders. Let* $E^n := E^n(d) := \{e \in E(d) \mid \nu(e) \in (S(d))^n\}$.
8. *A function,* $\kappa := \kappa_d : S(d) \cup E(d) \to \mathcal{O}^* \cup \mathcal{P}$ *which returns for each spider* $s \in S(d)$ *its label* $\kappa(s) \in \mathcal{O}^*$, *and which returns for each predicate edge* $e \in E(d)$ *its label* $\kappa(e) \in \mathcal{P}$, *where we have* $ar(\kappa(e)) = n$ *for* $e \in E^n(d)$ *(i.e.,* $\kappa(e)$ *has the appropriate arity).*

Define CONCEPTUAL SPIDER DIAGRAMS (CSD) *as follows:*

- *Every unitary conceptual spider diagrams is a conceptual spider diagram.*
- *if* $\mathcal{D}_1$ *and* $\mathcal{D}_2$ *are finite bags (multisets) of conceptual spider diagrams, then* $\vee(\mathcal{D}_1 \uplus \mathcal{D}_2)$ *and* $\wedge(\mathcal{D}_1 \uplus \mathcal{D}_2)$ *are conceptual spider diagrams, where* $\uplus$ *is the union for bags.*

Note that there is an empty CSD, $\bot := (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$. To provide an example for Def. 2, we return to the diagram of Fig. 6. For the types, we use to following abbreviations: A(nimal), C(at), T(hing), M(at), R(ug). The abstract syntax of the concrete diagram of Fig. 6 is given by:

$$
\begin{aligned}
L(d) &:= \{A, C, T, M, R\} \\
Z(d) &:= \{(\{A\}, \{C, T, M, R\}),\ (\{A, C\}, \{T, M, R\}),\ (\{T\}, \{A, C, M, R\}), \\
&\qquad (\{T, M\}, \{A, C, R\}),\ (\{T, R\}, \{A, C, M\}),\ (\{\}, \{A, C, T; M; R\})\} \\
Z^*(d) &:= \{(\{A, C\}, \{T, M, R\})\} \\
S(d) &:= \{s_1, s_2, s_3, s_4\} \\
\eta_d(s_1) &:= \eta_d(s_2) := \{(\{A, C\}, \{T, M, R\})\} \\
\eta_d(s_3) &:= \{(\{T, M\}, \{A, C, R\}),\ (\{T, R\}, \{A, C, M\})\} \\
\eta_d(s_4) &:= \{(\{T, M\}, \{A, C, R\})\} \\
E(d) &:= \{e_1, e_2\} \\
\nu_d(e_1) &:= (s_1, s_3) \text{ and } \nu_d(e_2) = (s_2, s_4) \\
\kappa_d(s_1) &:= \kappa_d(s_3) := \kappa_d(s_4) := *,\ \kappa_d(s_2) := \text{Yoyo},\ \kappa_d(e_1) := \kappa_d(e_2) := \text{on}.
\end{aligned}
$$

The semantics is based on classical Tarski-style interpretations.

Definition 3 (Interpretation). *An* INTERPRETATION *is a pair* $\mathcal{I} := (U, I)$ *consisting of an* UNIVERSE *and an* INTERPRETATION MAPPING I *which maps object names to elements of* U, *concept names to subsets of* U, *and predicate names of arity* n *to subsets of* U^n.

A point to note is that we are conforming to the general semantics of SDs, but contrasting to the usual semantics of FOL and CGs, by allowing *empty* universes. We are now prepared to provide the semantics for CSDs, and in fact we provide two slightly different readings of CSDs: the SD-semantics, where different spiders denote different objects; and the VP-semantics, where different spiders might denote the same object.

Definition 4 (Semantics). *Let a unary CSD* $d := (L, Z, Z^*, S, E, \nu, \kappa)$ *over an alphabet* $\mathcal{A} := (\mathcal{O}, \mathcal{C}, \mathcal{P})$ *and an interpretation* $\mathcal{I} := (U, I)$ *be given.*

- *First, we canonically extend* I *to zones* (a, b) *and regions* r *by setting*

$$I(a,b) := \bigcap_{C \in a} I(C) \cap \bigcap_{C \in b} \overline{I(C)} \qquad \text{and} \qquad I(r) := \bigcup_{z \in r} I(z)$$

- *We say that the* PLANE TILING CONDITION *holds iff we have* $\bigcup_{z \in Z} I(z) = U$ *(this condition reflects the Euler diagram features of CSDs).*
- *Any mapping* $val : S \to U$ *with* $val(s) = I(\kappa(s))$ *for* $\kappa(s) \neq *$ *is called* VALUATION *(of the spiders). If we have* $val(s) \in I(\eta(s))$ *for each spider* $s \in S$*, we say that* val *satisfies the* SPIDERS CONDITION*. If* val *is injective, we say that* val *satisfies the* STRANGERS CONDITION*. If for each edge* $e \in E$ *with* $\nu(e) = (s_1, \ldots, s_n)$*, we have* $(val(s_1), \ldots, val(s_n)) \in I(\kappa(e))$ *we say that* val *satisfies the* PREDICATES CONDITION*.*
- *We say that* $\mathcal{I}$ SATISFIES d IN THE VP-SENSE *iff the plane tiling condition holds and if there exists a valuation which satisfies the spiders condition and the predicates condition. We write* $\mathcal{I} \models_{VP} d$*. If the valuation additionally satisfies the strangers condition then* $\mathcal{I}$ SATISFIES d IN THE SD-SENSE*, and we write* $\mathcal{I} \models_{SD} d$*.*

6 Conclusion

We have provided a novel formal diagrammatic system called conceptual spider diagrams which utilizes useful features from CGs and SDs. A demonstration of its potential has also been provided via examples. This is a step towards a unification of CGs and SDs, and this "best of breed" approach looks promising. It also raises many interesting avenues of future research.

Firstly, the usability of the system of CSDs has to be thoroughly compared to SDs and to CGs. Although the examples in section 4 give an indication that that CSDs are easier to read than CGs, this benefit might get lost if we have type hierarchies with many overlapping concepts, since this could lead to underlying Euler diagrams where the contours are mutually intersecting to a large extent (thereby rendering the diagrams more cluttered [17] and hence more difficult to read). One would imagine that the more disjointness-constraints a type-hierarchy has the easier the corresponding Euler diagram is to read, but the associated tradeoff needs to be scrutinized carefully.

Secondly, it has to investigated whether CSDs can be extended in a way that not only the type-hierarchy of concepts, but the type-hierachy of relations can be diagrammatically depicted as well in a convenient way.

Thirdly, sound and complete calculi for the system have to be developed. Since we have two different semantics, we will require two slightly different calculi: these calculi should have many rules in common, differentiating only in rules which reflect the difference between the two semantics. Since CSDs are a hybrid of SDs and CGs, it is desirable that the calculi do not differ too much

from the existing calculi for SDs and CGs.[3] Furthermore, if CSDs are to be used in practice it is likely that only limited subsets of all the concepts in a given type-hierarchy are going to used at any one time (as we have shown in the right-hand diagram of Fig. 5). As an example of the types of rules for CSDs, recall that in the calculi for SDs, there are rules which allow the addition or removal of contours; it would be reasonable to have rules in the CSD calculi which allow the addition or removal of contours *with respect to a given type-hierarchy*. Another thing to keep in mind is that a set of rules designed with proving completeness of the system in mind might differ dramatically from a set of rules designed for human interaction. Trying to capture all of these features means that the rules for the calculi will have to designed very carefully.

In the long term the intention is to develop CSDs as a fully fledged, formal, diagrammatic reasoning system, unifying the existing systems of SDs, CDs and CGs, utilising the most appropriate features of each notation.

References

[1] Frithjof Dau. Types and tokens for logic with diagrams: A mathematical approach. In Karl Erich Wolff, Heather D. Pfeiffer, and Harry S. Delugach, editors, *Conceptual Structures at Work*, volume 3127 of *LNCS*, pages 62–93. Springer, Berlin – Heidelberg – New York, 2004.

[2] Frithjof Dau. Formal, diagrammatic logic with conceptual graphs. In Pascal Hitzler and Henrik Scharfe, editors, *Conceptual tructures in Practice*. CRC Press (Chapman and Hall/Taylor & Francis Group, 2008.

[3] L. Euler. Lettres a une princesse dallemagne sur divers sujets de physique et de philosophie. *Letters*, 2:102–108, 1775. Berne, Socit Typographique.

[4] A. Fish and J. Flower. Investigating reasoning with constraint diagrams. In *Visual Language and Formal Methods 2004*, volume 127 of *ENTCS*, pages 53–69, Rome, Italy, 2005. Elsevier.

[5] Andrew Fish, Jean Flower, and John Howse. The semantics of augmented constraint diagrams. *Journal of Visual Languages and Computing*, 16:541–573, 2005.

[6] J. Flower, A. Fish, and J. Howse. Euler diagram generation. *Accepted for the Journal of Visual Languages and Computing*, 2007.

[7] J. Flower, J. Masthoff, and G. Stapleton. Generating readable proofs: A heuristic approach to theorem proving with spider diagrams. In *Proceedings of 3rd International Conference on the Theory and Application of Diagrams*, volume 2980 of *LNAI*, pages 166–181, Cambridge, UK, 2004. Springer.

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

[9] Joseph Gil, John Howse, and Stuart Kent. Formalizing spider diagrams. In *IEEE Symposium on Visual Languages*, pages 130–137, 1999.

[10] C. Gurr. Effective diagrammatic communication: Syntactic, semantic and pragmatic issues. *Journal of Visual Languages and Computing*, 10(4):317–342, 1999.

[3] For the semantics in the SD-sense, a promising approach is to take one of the existing adaquate calculi for SDs, and augment it with rules which allow to generalize or even remove relations (such rules usually appear in calcusli for simple CGs, as they have been developped by Prediger, Dau, or Chein and Mugnier).

[11] E. Hammer and S. J. Shin. Euler's visual logic. *History and Philosophy of Logic*, pages 1–29, 1998.
[12] Weiss Hartshorne and Burks, editors. *Collected Papers of Charles Sanders Peirce*, Cambridge, Massachusetts, 1931–1935. Harvard University Press.
[13] J. Howse, F. Molina, S-J. Shin, and J. Taylor. On diagram tokens and types. In *Proceedings of 2nd International Conference on the Theory and Application of Diagrams*, pages 146–160, Georgia, USA, April 2002. Springer.
[14] J. Howse and S. Schuman. Precise visual modelling. *Journal of Software and Systems Modeling*, 4:310–325, 2005.
[15] J. Howse, G. Stapleton, and J. Taylor. Spider diagrams. *LMS Journal of Computation and Mathematics*, 8:145–194, 2005.
[16] John Howse, Gem Stapleton, and John Taylor. Spider diagrams. *LMS Journal of Computation and Mathematics*, 8:145–194, 2005.
[17] Chris John, Andrew Fish, John Howse, and John Taylor. Exploring the notion of clutter in Euler diagrams. In *4th International Conference on the Theory and Application of Diagrams*, pages 267–282, Stanford, USA, 2006. Springer.
[18] Stuart Kent. Constraint diagrams: Visualizing assertions in object-oriented models. In *OOPSLA*, pages 327–341. ACM Press, 1997.
[19] 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.
[20] Jill H. Larkin and Herbert A. Simon. Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11(1):65–100, 1987.
[21] O. Lemon and I. Pratt. Spatial logic and the complexity of diagrammatic reasoning. *Machine GRAPHICS and VISION*, 6(1):89–108, 1997.
[22] 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.
[23] Atsushi Shimojima. *On the Efficacy of Representation*. PhD thesis, The Department of Philosophy, Indiana University, 1996.
[24] Sun-Joo Shin. *The Logical Status of Diagrams*. Cambridge University Press, 1994.
[25] Sun-Joo Shin. *The Iconic Logic of Peirce's Graphs*. Bradford Book, Massachusetts, 2002.
[26] John F. Sowa. *Conceptual structures: information processing in mind and machine*. Addison-Wesley, Reading, Mass., 1984.
[27] G. Stapleton, J. Howse, and J. Taylor. A decidable constraint diagram reasoning system. *Journal of Logic and Computation*, 15(6):975–1008, December 2005.
[28] G. Stapleton, J. Masthoff, J. Flower, A. Fish, and J. Southern. Automated theorem proving in Euler diagrams systems. *Journal of Automated Reasoning*, 2007.
[29] 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.
[30] Gem Stapleton. *Reasoning with Constraint Diagrams*. PhD thesis, Visual Modelling Group, Department of Mathematical Sciences, University of Brighton, 2004.
[31] N. Swoboda and G. Allwein. Using DAG transformations to verify Euler/Venn homogeneous and Euler/Venn FOL heterogeneous rules of inference. *Journal on Software and System Modeling*, 3(2):136–149, 2004.
[32] J. Venn. On the diagrammatic and mechanical representation of propositions and reasonings. *Phil.Mag*, 1880.
[33] VMG. The visual modeling group homepage, university of brighton. `http://www.cmis.brighton.ac.uk/Research/vmg/`.