Formalizing Spider Diagrams

Joseph (Yossi) Gil*
Department of Computer Science
Technion—IIT, Haifa 32000, Israel
yogi@cs.technion.ac.il

John Howse†
School of Computing & Mathematics
University of Brighton, Brighton, UK
John.Howse@brighton.ac.uk

Stuart Kent†
Computing Laboratory
University of Kent, Canterbury, UK
S.J.H.Kent@ukc.ac.uk

Abstract

Geared to complement UML and to the specification of large software systems by non-mathematicians, spider diagrams *are a visual language that generalizes the popular and intuitive Venn diagrams and Euler circles. The language design emphasized scalability and expressiveness while retaining intuitiveness. In this extended abstract we describe spider diagrams from a mathematical standpoint and show how their formal semantics in terms of logical expressions can be made. We also claim that all spider diagrams are self-consistent.*

Keywords Visual formalisms, software specification, formal methods.

1. Introduction

Circles or closed curves, which we will call contours, have been in use for the representation of classical syllogisms since at least the Middle Ages [11]. The Swiss mathematician Leonhard Euler (1707-1783) introduced the notation we now call *Euler circles* (or Euler diagrams) [1] to illustrate relations between sets. This notation uses the topological properties of enclosure, exclusion and intersection to represent the set-theoretic notions of subset, disjointness, and intersection, respectively.

The 19^{th} century logician John Venn used contours to represent logical propositions [16]. In Venn diagrams all contours must intersect. Moreover, for each non-empty subset of the contours, there must be a connected region of the diagram, such that the contours in this subset intersect at exactly that region. Shading is then used to show that a particular region of the resulting map is empty.

In 1896, the logician Charles Peirce augmented Venn diagrams by adding X-sequences as a means for denoting elements [14]. An X-Sequences connecting a number of "minimal regions" of a Venn-diagram, indicates that their union is not empty. Peirce also gave a mechanism for writing disjunctive information, which we will not discuss here.

As an indication of the popularity and intuitiveness of Venn-Peirce diagrams is the fact that they are used in elementary schools for teaching set theory as an introduction to mathematics. Still, only recently, full semantics and inference rules have been developed for Venn-Peirce diagrams [15] and Euler circles [4].

As a means for writing constraints on sets and their relationships with other sets, Venn-Peirce diagrams are expressive, but complicated to draw because all possible intersections have to be drawn and then some regions shaded. Drawing the Venn diagram of four or more sets is quite challenging. As shown by More [12] in the late fifties, there is an algorithm for adding a new contour to a Venn diagram. Although it is possible to do so indefinitely, the contours quickly assume weird shapes, with exponential increase in their curvature. The resulting diagram is very complicated and difficult to follow. Indeed, it is rare to see Venn diagrams of four or more contours. On the other hand, Euler circles are intuitive and easier to draw, but are not as expressive as Venn diagrams because they lack provisions for shading and for "X-Sequences".

In view of their relative merits, it seems natural to combine the two notations, by relaxing the demand that all curves in Venn-diagrams must intersect. Doing this and more are *spider diagrams*. They are named after the "spider shape" which generalizes X-sequences in that a "minimal region" may have more than one spider in it, e.g., to denote that a set has two or more elements.

*Work done in part while the author was at the IBM T. J. Watson Research Center

†Partially funded by the UK EPSRC, grant numbers GR/K67304 and GR/M02606

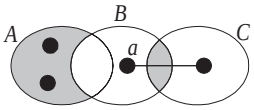

Fig. 1: An example spider diagram.

Fig. 1 is a simple spider diagram consisting of three spiders and three contours. The figure uses various visualization techniques to specify that

$$|A - B| = 2$$
$$C \cap A = C \cap B = \emptyset$$
$$a \in ((B - C) - A) \cup ((C - A) - B)$$

Spiders may be connected by strands or ties in a region, to indicate that elements denoted by the spiders may or must be the same in that region. Further, a region containing spiders may also be shaded to denote that there are no elements in that region other than those shown by the spiders. Thus, spider diagrams allow placing both an upper bound and a lower bound on the size of a set. Among other extensions to traditional notation we find in spider diagrams the notion of "projections" which are used to show the intersection of more than three curves in a clear and uncluttered manner.

Spider diagrams have emerged from a succession of attempts to provide the software designer with precise, yet intuitive tools to specify a system prior to actually coding it. Work along this line of research can be traced all the way back to Harel's seminal state-charts. It was only the sound mathematical foundation which they stand upon that enabled the development of "executing" CASE tools [6]. With the advent of the object-oriented paradigm, and the modernization of software design, came a series of notations, for example BON [17], OML [2], and the celebrated UML standard [13]. Most of these visual languages are quite restricted in their expressive power in lacking provisions for denoting first order predicate logic formulae, which are so essential for describing system invariants. Such formulae are written using either in-borrowed mathematical notations (as in BON), in an auxiliary specialized textual language designed for that purpose, e.g., OCL [18], or worse, as natural language annotations (as was the case with OML).

Spider diagrams have emerged from work on *constraint diagrams* [8], which were introduced as an attempt to remedy this situation. The constraint diagram in Fig. 2 expresses (amongst other constraints) an invariant on a model of a library system: for any library object, and any copy of that library which is on hold, that copy's publication must be the same as that associated with the reservation for which it is on hold

$$\forall x \in \mathsf{Libraries} \bullet \forall y \in x.\mathsf{collection} \cap \mathsf{OnHold} \bullet$$
$$y.\mathsf{OnHold.reserved} = y.\mathsf{publication}.$$

This reading is obtained by interpreting the *s as wildcards, universal quantifiers ranging over the regions which

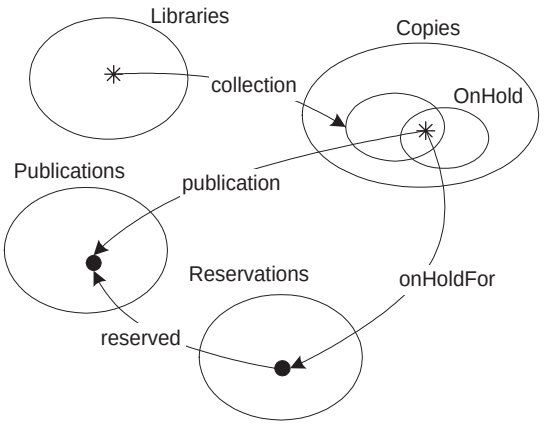

Fig. 2: A constraint diagram.

contain them, and arrows as showing the range of relations when their domain is restricted to the set or element at their source. Spider diagrams are used to show relationships between all the sets and elements involved. In fact, spider diagrams can be thought of as constraint diagrams without the arrows.

Constraint diagrams proved to be an intuitive and useful tool for design, and several extensions of these were proposed, including a variant designed for expressing pre- and post-conditions pairs [9], a three dimensional version for behavioral specification [3], and a version used for meta-specification, i.e., the specification of design patterns [10]. At the same time, constraint diagrams were employed in the software industry to produce informal descriptions of systems.

Spider diagrams, the most fundamental layer of constraint diagrams, are the subject of this treatise. Our efforts towards upgrading the language into a true "visual formalism" [5], include the discussion of the essential syntax, semantics and properties of spider diagrams, treating them both as topological and notational creatures. Following the work of Shin [15] and Hammer [4] we show how formal semantics in the form of logical formulae for spider diagrams can be given.

Beyond visual software modeling, spider diagrams have an impact on the field of diagrammatic reasoning, a relatively new field which looks at logic and logical reasoning from a wholly new perspective. Such extensions are discussed in [7].

Outline. The remainder of this paper is organized as follows. Sec. 2 defines the syntax, and provides an informal semantics which helps to motivate and explain the various syntactic elements. This section treats spider diagrams as topological creatures. A formal definition of spider diagrams as a notational device devoid of geometrical and topological undertones is given in Sec. 3. This definition is then used in Sec. 4 to provide formal semantics of the notation. Sec. 4 also quotes the main result in the study

Fig. 3. The three possible relationships between two contours: intersection (a), disjointness (b), and containment (c).

of spider diagrams, namely that all legal diagrams are consistent. Finally, Sec. 5 concludes the paper and describes future research. Due to space limitations proofs are omitted from this paper. Similarly, some definitions, which were overly technical, were abridged.

2. Syntax and Informal Semantics

2.1. Contours

Contours are shapes used in a spider diagram to denote sets. Formally, a *contour* is a closed non-self-intersecting plane curve.

Although it is convenient to draw contours as ovals, this is not mandatory. Other topologies may be used for making a visual distinction between different kinds of contour. For example, in object-oriented modeling, rectangle contours are used to indicate that a set corresponds to a class of objects. Since convex contours tend to be more visually pleasing we usually try to draw them as such.

Different iconic representations, or line styles, may be used to distinguish between different kinds of contours. In state-chart diagrams for example, thick lines may be used to denote initial states. We will also use dashed lines to denote projections, which are a special kind of contour. All concepts described in this section are independent of the chosen shape or the iconic representation of a contour.

A diagram contains at least one contour, called the *boundary contour*. The boundary is a contour which is not contained in, nor does it intersect with, any other contour. We do not usually bother to draw the boundary contour: it is assumed to be the bounding box for the diagram, be it the edges of the drawing surface, the edges of a figure, etc.

Excepting the boundary, all contours in a Venn diagram must intersect. We do not require this property in spider diagrams. In spider diagrams, just as in Euler circles, two contours can stand in one of three relations (Fig. 3). An *intersection* between the contours will mean that the sets they denote *may* intersect. Thus, with the absence of any other information, no statement is being made of the relationship between these two sets. If the contours do not intersect, then they are either *disjoint*, with the implication that their denoted sets are disjoint, or one of of them is *contained* in the other, with a corresponding implication on the relationship between the sets they denote.

There is a much greater variety of relationships among more than two contours. Consider for example Fig. 4, in which one contour is contained in the "union" of two others. The intended semantics is of course that $A \subseteq B \cup C$.

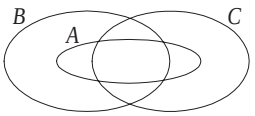

Fig. 4: One possible relationship between three contours..

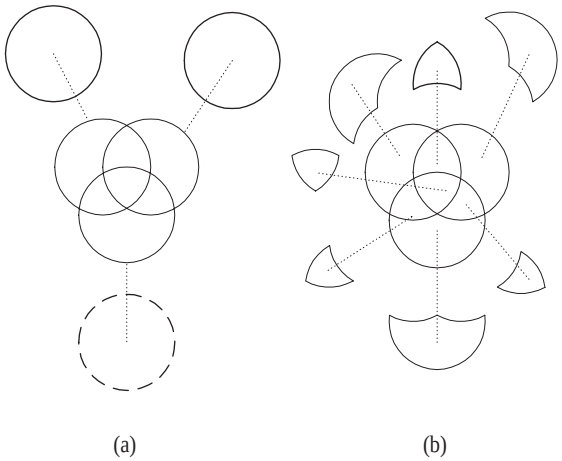

Fig. 5: Contours (a) and zones (b) in a clover.

Part of the challenge in giving precise semantics to spider diagrams is in systematically dealing with the general case of relationships among any number of contours.

A contour can be labeled. By convention, contour labels are initially capitalized.

2.2. Districts, Regions and Zones

The meaning of a diagram is obtained from the topology of the contours in it. The terms district, region, and zone will become pertinent in the study of this topology.

A *district* (sometimes called a *basic region*) is the set of points in the plane enclosed by, or lying on, a contour. The district of the boundary contour is called the *domain* since it consists of all points of the diagram. By definition, a district is a connected set. Also, since a district includes the points on the contour itself, topologically it is a closed set. Fig. 5a shows the districts of a *clover*—three mutually intersecting non-boundary contours.

The regions of a diagram are the non-empty sets which can be formed from its districts by means of set union, intersection and difference operations. More formally, a *region* has the following recursive definition: Any district is a region, and in addition, for any two regions r_1 and r_2, the set $r_1 \cup r_2$ is a region, as are the sets $r_1 \cap r_2$ and $r_1 - r_2$ provided they are not empty.

Regions are not necessarily closed, nor do they have to be connected. For example, the region corresponding to $(A - C) \cup (A - B)$ in Fig. 4 is not connected, neither it is a closed set, nor an open one.

We insist that the number of contours in a diagram is finite. It follows that there are regions which are minimal with respect to set containment. This special kind of region

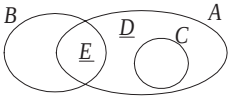

Fig. 6: An example of using zone labels.

is a called zone. Formally, a *zone* (or a *minimal region*) is a region having no other region as a proper subset. Notice that a zone does not have to be connected. For example, in Fig. 4, the zone corresponding to $(B \cap C) - A$ is not connected.

Fig. 5 shows all but one of the zones of the clover. The zone not shown is that which is formed by subtracting the districts of all non boundary contours from the domain.

It is not difficult to see that contours partition the domain into disjoint zones. Accordingly, we have that all the regions can be generated by taking the union of any non-empty collection of zones. The clover, for example, has $2^8 - 1 = 255$ regions in total, which is the number of non-empty ways its eight zones can be combined.

A zone can also be a district, as it the case for two out of the three zones of Fig. 3b. However, it follows from the definition that a zone is either completely contained in a district, or it disjoint from it. This dichotomy suggests a canonical method for denoting zones.

Definition 1 *Let $\mathcal{C}$ be a set of contours, and let $b \in \mathcal{C}$ be a special boundary contour. Then, the pair $\langle C_1, C_2 \rangle$ is called a* contour division *if $b \in C_1$, $C_1 \cap C_2 = \emptyset$ and $C_1 \cup C_2 = \mathcal{C}$.*

For a zone z, let $C^+(z)$ be the set of contours that contain it, and let $C^-(z)$ be the set of contours that don't. Then, $\langle C^+(z), C^-(z) \rangle$ is a contour division of the contours of the diagram which uniquely identifies z. The converse, namely that all contours divisions correspond to zones is true in Venn diagrams, but may not hold in spider diagrams.

Topologically, a zone z is the intersection of the districts of $C^+(z)$, minus the union of districts of $C^-(z)$. For example, the zone highlighted in Fig. 5b is calculated by taking the intersection of the sets denoted by the contours highlighted with solid border in Fig. 5a and subtracting the set denoted by the contour highlighted with a dashed border.

Zones can be optionally labeled. By convention a zone label is underlined, initially uppercase and placed inside its zone. Using this convention we can read now read Fig. 6 and determine that its semantics is (among other things)

$$A - (D \cup C) = A \cap B = E.$$

The spider diagram Fig. 7 summarizes (in a reflective fashion) some of the terms introduced above: regions are sets of points, districts and zones are regions, and there might be zones which are also districts.

2.3. Spiders

So far, our expressive power was limited to sets and their relationships. In order to be able to make statements about set

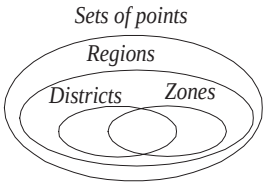

Fig. 7. The set-theoretical relationship between zones, districts and regions.

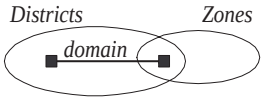

Fig. 8: The spider of the domain.

members, we need a new shape: the spider, which is used to denote elements. Spiders are similar to X-sequences, or chains, in Peirce diagrams, except that unlike X-sequences, spiders must be distinct (unless they are joined by a *tie* or by a *strand*, see below).

The visual representation of a spider is as a tree with nodes, called *feet*. A foot is a drawn as a little black circle or square, and the connecting edges, the *legs* of the spider, are drawn as straight lines. We chose the tree representation instead of the linear structure of X-sequence used in traditional Venn-Peirce diagrams since we found that the greater flexibility enables more visually pleasing diagrams.

We say that a spider *touches* a zone if it has a foot in that zone. No spider may touch the same zone twice. A spider *inhabits* the region formed by the union of all the zones that the spider touches. This region, called the *habitat* of the spider, is intuitively where the element denoted by the spider might reside.

Spiders can be labeled. By convention, spider labels are all lower case. As a simple reflective example, Fig. 8 shows that the domain is a district, but it may be in addition a zone (this happens if a diagram has no contours other than the boundary).

There is a slight semantical difference between spiders with circles as feet and spiders with squares as feet. A spider whose feet are circles corresponds to existential quantification. Thus, in Fig. 1, the semantics of the two left most spiders is that there are two distinct anonymous elements in $A - B$. A spider whose feet are squares represents a given element. The semantics of Fig. 8 is that the *specific* element called "domain" is a member of the set "Districts".

A more elaborate example is given in Fig. 9. The semantics is that there exists *a*, *b*, *c* such that

$$b \in (B - C) \cup (C - A)$$
$$a \in (A - B) - C$$
$$c \in (U - A) - B$$

where U is the universal set denoted by the boundary contour. In addition, the semantics includes the condition that

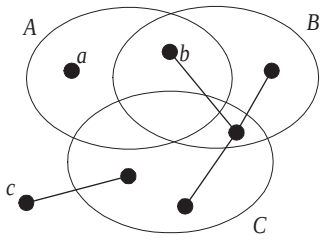

Fig. 9: Three spiders in a clover example.

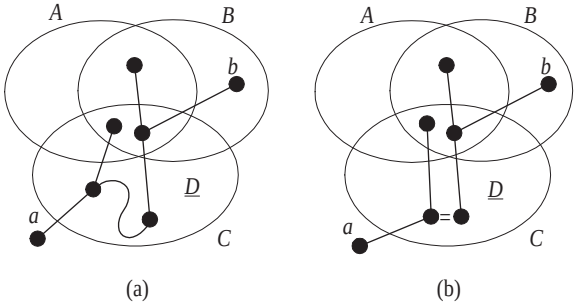

Fig. 10: Strands (a) and ties (b) in a spider diagram.

all elements are distinct, i.e., $a \neq b$, $b \neq c$, and $a \neq c$.

2.4. Strands and Ties

We introduce the notion of *strands* and *ties* to provide a means for denoting that spiders may (or must) be the same should they occur in a certain zone.

The increased expressivity does not come at the cost of visual clarity. There is an intuitive and concise iconic representation for both strands and ties. Suppose that nodes of two spiders placed in the same zone, are connected by a "strand", drawn as a wavy line, as shown in Fig. 10a. Then, this means that the elements that these spiders denote may be the same if they occur in this zone. In Fig. 10a elements a and b are required to be distinct only if they are not members of the zone D.

Dually, if the same two nodes are connected by a "tie", drawn as a double straight line resembling an equal sign (see Fig. 10b for an example), then the two elements *must* be the same if they occur in the same zone. Thus, the semantics of Fig. 10b is also that

$$(a \in D \wedge b \in D) \Rightarrow a = b.$$

The *nest* of spiders *s* and *t* is the union of those zones *z* having the property that there is a sequence of spiders

$$s = s_0, s_1, s_2, \ldots, s_n = t$$

such that, for $i = 0, \ldots, n-1$, s_i and s_{i+1} are connected by a tie in z. Two spiders which have a non-empty nest are referred to as *mates*. If both the elements denoted by spiders s and t are in the set denoted by the same zone in the nest of s and *t*, then s and t denote the same element.

A *strand* is a wavy line connecting two feet, from different spiders, placed in the same zone. The *web* of spiders s and t is the union of zones z having the property that there is a sequence of spiders

$$s = s_0, s_1, s_2, \ldots, s_n = t$$

such that, for $i = 0, \ldots, n-1$, s_i and s_{i+1} are connected by a tie or by a strand in z. So the *nest* of spiders *s* and *t* is a subregion of the *web* of spiders s and t. Two spiders s and t may (but not necessarily must) denote the same element if that element is in the set denoted by the web of s and t. Clearly, if there is a tie between feet, then a strand between those feet is redundant. Similarly, multiple strands or ties between the same pairs of feet are redundant.

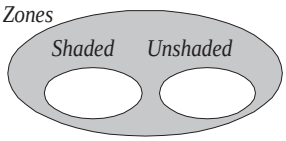

Fig. 11: Shaded and unshaded zones.

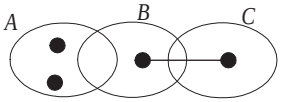

Fig. 12: Specifying cardinality with spiders.

2.5. Shading and Schrödinger Spiders

Venn-Peirce diagrams use shading to specify that a zone is empty. Hence, a shaded zone in Venn-Peirce diagrams may not contain an X-sequence. This condition is relaxed in spider diagrams, and a shaded zone may contain spiders. The semantics of a *shaded zone* is that the set it denotes may not contain elements other than those indicated by the spiders which touch that zone. A shaded zone which has no spiders is thus empty, in agreement with Venn-Peirce notation. As shown in Fig. 11, zones are either *shaded* or *unshaded*.

Just like labels, shading is not technically part of the geometry of a diagram. They are rather a property of a set of its points. It is still useful to render shaded zones by actually shading them as in Fig. 11. Although visually appealing, rendering a zone shaded is difficult to draw freehand. Another alternative for a visual indication of shading is placing a $\times$ symbol in the zone.

Spiders can be used to place a lower bound on the number of elements in a set. In Fig. 12, we have that

$$|A - B| \geq 2$$
$$|(B - A) - C) \cup (C - B)| \geq 1.$$

Shading a zone which includes spiders has the effect of placing an upper bound on the cardinality of elements in the set denoted by that zone. In Fig. 13a, the set A contains exactly 2 elements; the two spiders in the zone mean lower-bound of 2 on its size, the shading of the zone ensures that this is also an upper-bound. Also, B contains at most 3 elements; it may contain less as the elements denoted by

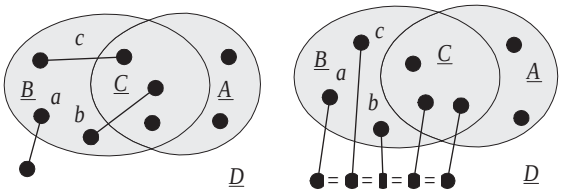

Fig. 13. Specifying cardinality with shading and spiders (a) and the effect of ties (b).

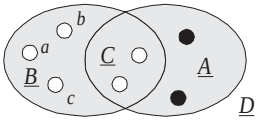

Fig. 14: Using Schrödinger spiders.

spiders a, b and c may be selected from other zones. In the same fashion we have that $1 \leq |C| \leq 3$.

In Fig. 13a, $|B|$ and $|C|$ are related: the more elements in B the less in C, and vice-versa. To avoid this dependency, an exclusive element could be introduced into the universal set, i.e., an element which is not a member of any of the sets represented by the other contours. In Fig. 13b, the same restrictions on $|B|$ and $|C|$ are in force, but this time if a, b and c denote elements in B this has no impact on on $|C|$, as the habitats of these spiders do not include C. The price is the introduction of the constraint that $|D| \geq 1$ when $|B| = |C| = 0$.

Besides being awkward to draw, and being less than immediately intuitive, the exclusive element of the universal set is undesirable since it forces an unnatural constraint on a diagram. We fix the situation by introducing a new symbol, the *Schrödinger spider*, which denotes a set whose size is either zero or one. Just like Schrödinger's cat one is not sure whether the element exists or not. Formally, a *Schrödinger spider* is nothing but a kind of a spider, whose semantics is specialized.

The feet of a Schrödinger spider are rendered differently from normal non-Schrödinger spiders. An example is given in Fig. 14, in which zone B contains three Schrödinger spiders, labeled a, b, and c, while region C contains two unlabeled Schrödinger spiders. In comparison with Fig. 13b, Fig. 14 is less cluttered, and does not force an element into zone D.

There is a limit to the amount we can express in this notation about the cardinality of sets. For example, we are unable to say that $|A| = |B|$ for a disjoint sets A and B. In order to retain the intuitiveness of spider diagrams, constraints such as this should be placed in an auxiliary textual annotation, not burdening the visual notation.

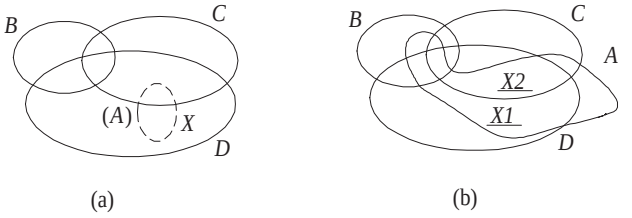

Fig. 15: A projected set (a) and its semantics (b).

2.6. Projections

Sometimes it is necessary to show a set in a certain context. Intersection can be used for just this purpose: an intersection of A and B shows the set A in the context of B and vice-versa. However, intersections also introduce regions which may not be of interest. Projections are equivalent to taking the intersection of sets, except that they introduce fewer regions, with the effect that regions which are not the focus of attention are not shown resulting in less cluttered diagrams.

A *projection* is a contour which is used to denote an intersection of a set with a "context". By convention, we use dashed iconic representation to make the distinction between projections and other contours.

A *determining label* must be associated with any projection p. This label is used to denote the set which is being projected. The convention is that determining labels are rendered within parenthesis when drawn in a diagram. A projection can also have a contour label.

Consider Fig. 15(a) for example. The dashed contour, labeled X, denotes the set obtained by "projecting" the set A onto the context $(D \cup C) - B$, i.e.,

$$X = A \cap ((D \cup C) - B)$$

The same semantics could have been obtained by using More's algorithm [12] to draw the Venn diagram of four contours, as in Fig. 15(b)

$$X = X_1 \cup X_2.$$

The simplicity of Fig. 15(a) when compared to Fig. 15(b) is self evident.

As noted above, the semantics of a projection is determined by its context, defined by:

Definition 2 *The* context *of a contour c, denoted $\kappa(c)$ is the smallest region that strictly contains the district of c.*

Strictness ensures that the district itself is not the region containing the contour. A projection p denotes the set obtained by intersecting the set denoted by its determining label with the set denoted by $\kappa(p)$. It must be possible to calculate the set denoted by $\kappa(p)$ from the sets denoted by contours other than p itself.

There are fascinating mathematical intricacies involving the intersection of several projections. Due to space constraints, we will not be able to discuss these here, nor shall

we be able to give the full semantics or even the exact definition of projections.

3. Formal Syntax of Spider Diagrams

In this section we give spider diagrams a formal definition which is independent of any topological and visual representations thereof. For space reasons, the definition of projections is omitted from this paper.

A *spider diagram* is a tuple

$$\langle \mathcal{C}, \beta, \mathcal{Z}, \mathcal{Z}^*, \mathcal{R}, \mathcal{S}, \mathcal{S}^*, \eta, \tau, \zeta, \rangle$$

whose components are defined as follows:

(i) $\mathcal{C}$ is a finite set whose members are called *contours*. The element β, which is not a member of $\mathcal{C}$, is called the *boundary contour*.

(ii) The set $\mathcal{Z} \subseteq 2^{\mathcal{C}}$ is the set of *zones*, while $\mathcal{Z}^* \subseteq \mathcal{Z}$ is the set of *shaded zones*. A zone $z \in \mathcal{Z}$ is *incident* on a contour $c \in \mathcal{C}$ if $c \in z$.

(iii) $\mathcal{R}$ is the set of *regions*, and let $\mathcal{R}' = \mathcal{R} \cup \emptyset$.

(iv) $\mathcal{S}$ is a set of *spiders*, while $\mathcal{S}^* \subseteq \mathcal{S}$ is a set of *Schrödinger spiders*.

(v) The function $\tau : \mathcal{S} \to \mathcal{R}$ returns the habitat of a spider. The function $\tau : \mathcal{S} \times \mathcal{S} \to \mathcal{R}'$ returns the nest of any two spiders, while $\zeta : \mathcal{S} \times \mathcal{S} \to \mathcal{R}'$ is a function that returns the web of any two spiders.

In addition, we use the value $\perp$, which is not a member of any of the sets we mention, to denote undefined values.

We use the dot notation to extract components of a tuple. Thus, $d.\mathcal{Z}$ denotes the zones of a diagram d.

A *spider multi-diagram* is a collection $\mathcal{D}$ of spider diagrams.

4. Formal Semantics

A *model* for a spider diagram

$$d = \langle \mathcal{C}, \beta, \mathcal{Z}, \mathcal{Z}^*, \mathcal{R}, \mathcal{S}, \mathcal{S}^*, \eta, \tau, \zeta, \rangle$$

is a triple

$$m = \langle \mathbf{U}, \psi, \Psi \rangle$$

such that $\mathbf{U}$ is a set and ψ and Ψ are functions:

- $\psi : \mathcal{S} \to \mathbf{U} \cup \{\perp\}$ maps given spiders to elements of $\mathbf{U}$ or to the special symbol $\perp$, and
- $\Psi : \mathcal{C} \to 2^{\mathbf{U}}$ maps contours to subsets of $\mathbf{U}$.

Given a diagram, we will define a predicate $P_d(\cdot)$, called its *semantics predicate*. A model *complies* with a diagram d if it satisfies its semantics predicate, i.e., $P_d(\mathbf{U})$ is true. A diagram is *consistent* if it has a compliant model. As shall become easy to see, the empty set is compliant with any legal diagram d which has no spiders, i.e., $d.\mathcal{S} = \emptyset$. Our main result is

Theorem 1 *All spider diagrams are self-consistent.*

The proof is based on the construction of the topological model of a spider diagrams. The details are omitted.

The definition of a model and compliance have a straightforward extension to a spider multi-diagram $\mathcal{D}$. The semantics predicate $P_{\mathcal{D}}(\mathbf{U})$ is simply the conjunction of the semantics predicate of the individual diagrams:

$$P_{\mathcal{D}}(\mathbf{U}) = \bigwedge_{d \in \mathcal{D}} P_d(\mathbf{U}).$$

Thm. 1 does not extend to multi-diagrams. The reason is that it is possible to simultaneously denote that a certain set is empty in one diagram and non-empty in another diagram.

Since non-given spiders are existentially quantified, we must map them into formal variables. A non-given spider is mapped into a formal variable which will existentially range over $\mathbf{U}$.

Let $s_1, \ldots, s_k$ denote the non-given non-Schrödinger spiders, and let $s_{k+1}, \ldots, s_{k+l}$ denote the non-given Schrödinger spiders. Let $\mathbf{U}' = \mathbf{U} \cup \{\perp\}$, then, predicate $P_d(\mathbf{U})$ is

$$\exists x_{s_1} \in \mathbf{U} \bullet \cdots \bullet \exists x_{s_k} \in \mathbf{U} \bullet \\ \exists x_{s_{k+1}} \in \mathbf{U}' \bullet \cdots \bullet \exists x_{s_{k+l}} \in \mathbf{U}' \bullet Q$$

where Q is a predicate involving no quantifiers defined by

$$Q = Q_1 \wedge Q_2 \wedge Q_3 \wedge Q_4 \wedge Q_5 \wedge Q_6$$

and the predicates Q_i, $i = 1, \ldots, 6$ are expanded below.

The function Ψ maps the contours of d to subsets of $\mathbf{U}$. Let us extend its definition to include the interpretation of other elements of d which are to denote sets in the model:

(i) *Boundary.* The boundary denotes the universal set.
$$\Psi(\beta) = \mathbf{U}$$

(ii) *Zones.* The semantics of a zone $z \in \mathcal{Z}$ is determined by which contours enclose it and which don't
$$\Psi(z) = \bigcap_{c \in z \vee c = \beta} \Psi(c) - \bigcup_{c \in \mathcal{C} - z} \Psi(c)$$
By letting the set intersection operation range over the boundary contour, we make sure that even the zone that is external to all contours has a well-defined semantics.

(iii) *Regions.* The value of Ψ of a region, or more generally any other collection of zones is simply the union of the semantics of the zones in the collection: For $r \in 2^{\mathcal{Z}} - \{\emptyset\}$, let
$$\Psi(r) = \bigcup_{z \in Z(r)} \Psi(z)$$
where, for any region r, $Z(r)$ is the set of zones contained in r.

Predicate Q_1 (the *plane tiling condition*) ensures that all elements fall within sets denoted by zones:

$$\bigcup_{z \in \mathcal{Z}} \Psi(z) = \mathbf{U}$$

This predicate, means that that an intersection of contours that doesn't appear as a zone must be empty. From it follows our intuitive interpretation of disjoint contours and containing contours: Let c_1 and c_2 be two distinct contours in a topological diagram. Then, it follows from the plane tiling condition that if c_1 is disjoint from c_2, then $\Psi(c_1) \cap \Psi(c_2) = \emptyset$. Similarly, if c_1 is contained in c_2 then it follows from that condition that $\Psi(c_1) \subseteq \Psi(c_2)$.

Predicates Q_2 (the *spider condition*) and Q_3 (the *Schrödinger spider condition*) ensure that an element denoted by a spider is in the set denoted by the habitat of the spider:

$$\begin{aligned} Q_2 &= \bigwedge_{s \in \mathcal{S} - \mathcal{S}^*} \psi(s) \in \Psi(\eta(s)) \\ Q_3 &= \bigwedge_{s \in \mathcal{S}^*} \psi(s) \in \Psi(\eta(s)) \cup \{\bot\} \end{aligned}$$

Predicate Q_4 (the *strangers condition*) ensures that elements denoted by two distinct spiders are equal then they must fall within the set denoted by their web:

$$\bigwedge_{s,t \in \mathcal{S}, s \neq t} \psi(s) = \psi(t) \Rightarrow \psi(s) = \bot \vee \psi(s) \in \Psi\left(\zeta(s,t)\right)$$

Predicate Q_5 (the *mating condition*) ensures that if the elements denoted by two distinct spiders fall within the set denoted by their nest, then these elements must be equal:

$$\bigwedge_{s,t \in \mathcal{S}} \bigwedge_{z \in Z(\tau(s,t))} \psi(s), \psi(t) \in \Psi(z) \Rightarrow \psi(s) = \psi(t)$$

Finally, predicate Q_6 (the *shading condition*) maintains that the set denoted by a shaded zone contains no elements other than those denoted by spiders

$$\bigwedge_{z \in \mathcal{Z}^*} \Psi(z) \subseteq \bigcup_{s \in \mathcal{S}} \{\psi(s)\}$$

Here we adopt the standard convention that a union over an empty range results in the empty set. Together with the spider condition, Q_6 ensures that the only elements in a set denoted by a shaded zone are the elements represented by any spiders impinging on that zone. Specifically, the set denoted by a shaded zone not containing feet of any spiders is empty.

5. Conclusions and Further Work

This is the first of a two part semantics for constraint diagrams. The second part will deal with the semantics of arrows. We have begun to explore rules for reasoning directly with the diagrams, building on recent work in reasoning with Venn diagrams and Euler circles. Early results look promising, see [7] for details. It is our eventual aim to develop tools to support conceptual modeling, including the modeling of software, based on these formal results. The formal work goes hand in hand with attempts to popularize and obtain feedback on the utility of the notation.

References

[1] L. Euler. *Lettres a Une Princesse d'Allemagne*, volume 2. 1761. Letters No. 102–108.

[2] D. Firesmith and B. Henderson-Sellers. Open Modeling Language (OML) notation specification. By the OPEN Consortium, 1996.

[3] J. Gil and S. Kent. Three dimensional software modelling. In *Proceedings of International Conference in Software Engineering*, Kyoto, Japan, May 1998. IEEE Press.

[4] E. Hammer. *Logic and Visual Information*. CSLI Publications, 1995.

[5] D. Harel. On visual formalisms. In J. Glasgow, N. H. Narayanan, and B. Chandrasekaran, editors, *Diagrammatic Reasoning*, pages 235–271. MIT Press, 1998.

[6] D. Harel, H. Lachover, A. Naamad, A. Pnueli, M. Politi, R. Sherman, A. Shtull-Trauring and M. Trakhtenbrot. Statemate: A working environment for the development of complex reactive systems. *IEEE Transactions on Software Engineering*, 16(4), 403-414, 1990.

[7] J. Howse, F. Molina, J. Taylor, S. Kent. Reasoning with Spider Diagrams. In *Proceedings of IEEE Symposium on Visual Languages* (VL99), IEEE Press, 1999.

[8] S. Kent. Constraint diagrams: Visualising invariants in object oriented models. In *In Proceedings of OOPSLA97*, ACM SIGPLAN Notices 32, 1997.

[9] S. Kent and Y. Gil. Visualising action contracts in oo modelling. *IEE Proceedings Software*, 1998. 145.

[10] A. Lauder and S. Kent. Precise visual specification of design patterns. In *Proceedings of ECOOP98*, Springer Verlag, 1998.

[11] R. Lull. *Ars Magma*. Lyons, 1517.

[12] T. More. On the construction of Venn diagrams. *Journal of Symbolic Logic*, 24, 1959.

[13] OMG. UML 1.1. specification. OMG Documents ad970802-ad970809, 1997.

[14] C. Peirce. *Collected Papers*. Harvard Univ. Press, 1933.

[15] S.-J. Shin. *The Logical Status of Diagrams*. CUP, 1994.

[16] J. Venn. On the diagrammatic and mechanical representation of propositions and reasonings. *Phil.Mag.*, 1880. 123.

[17] K. Walden and J.-M. Nerson. *Seamless Object-Oriented Software Architecture: Analysis and Design of Reliable Systems*. Object-Oriented Series, Prentice-Hall, 1994.

[18] J. Warmer and A. Kleppe. *The Object Constraint Language: Precise Modeling with UML*. Addison-Wesley, 1998.