Enhancing the Expressiveness of Spider Diagram Systems

Gem Stapleton and John Howse
Visual Modelling Group
University of Brighton
Brighton, UK
www.cmis.brighton.ac.uk/research/vmg
{g.e.stapleton,john.howse}@brighton.ac.uk

Abstract

Many visual languages based on Euler diagrams have emerged for expressing relationships between sets. The expressive power of these languages varies, but the majority are monadic and some include equality. Spider diagrams are one such language, being equivalent in expressive power to monadic first order logic with equality. Spiders are used to represent the existence of elements or specific individuals and distinct spiders represent distinct elements. Logical connectives are used to join diagrams, increasing the expressiveness of the language. Spider diagrams that do not incorporate logical connectives are called unitary diagrams. In this paper we explore generalizations of the spider diagram system. We consider the effects of these generalizations on the expressiveness of unitary spider diagrams and on conciseness.

1 Introduction

Recent times have seen various formal diagrammatic logics and reasoning systems emerging [6, 13, 17, 25, 26, 31, 33]. Many of these logics are based on the popular and intuitive Euler diagrams augmented with shading. The diagrams in figure 1 are all based on Euler diagrams.

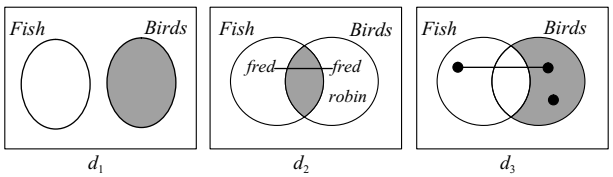

Figure 1. Various extended Euler diagrams.

The diagram d_1 expresses that there are no birds (by the use of shading) and that nothing is both a fish and a bird (since the two circles have disjoint interiors). The diagram d_2 is an Euler/Venn diagram [31] and expresses that nothing is both a fish and a bird (by the use of shading) and there is something called $fred$ that is either a fish or a bird, but not both. Furthermore, $robin$ is a bird but not a fish. Note that $fred$ and $robin$ could represent the same individual. Euler/Venn diagrams use shading to express the emptiness of a set and *constant sequences* to make statements about specific individuals. Finally, d_3 is a spider diagram [16] and expresses that something is either a fish or a bird but not both and there is at least one but at most two elements in the set $Birds - Fish$ (by the use of *existential spiders* and shading). So, by contrast to Euler/Venn, spider diagrams use shading to place upper bounds on set cardinality and spiders place lower bounds on set cardinality. Whilst the spiders in d_3 represent the existence of elements, spider diagrams use *constant spiders* to make statements about specific individuals.

Others have introduced diagrammatic logics based on Venn diagrams which form, essentially, a fragment of the Euler diagram language. Peirce used $\otimes$-sequences to assert non-emptiness and, instead of shading, o-sequences to assert emptiness [23]. The diagram d_1 in figure 2 is a Venn-Peirce diagram and expresses that $Fish - Birds = \emptyset$ or $Birds - Fish \neq \emptyset$. This example illustrates a key difference between the use of shading and the use of o-sequences because the statement made is disjunctive. The diagram d_2 is a Venn-II diagram [25] and uses an $\otimes$-sequence to express that $Fish \neq \emptyset$ and shading to express $Birds - Fish = \emptyset$. Venn-II does not incorporate o-sequences.

The expressiveness of various diagrammatic logics has been established. The Venn-II system has been shown to be equivalent in expressive power to monadic first order logic (without equality) [25]. In monadic first order logic, all of the predicate symbols are unary and they correspond to contour labels. So, for example, Venn-II cannot express that a property holds for a unique individual. Spider diagrams properly increase expressiveness over Venn-II. In [30] it is

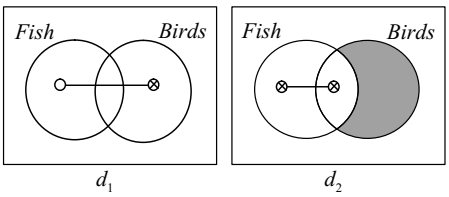

Figure 2. Venn-Peirce and Venn-II diagrams.

shown that the language of spider diagrams (without constant spiders) is equivalent in expressive power to monadic first order logic with equality and, hence, can express that a property holds for a unique individual. Furthermore, it has been shown that the inclusion of constant spiders does not increase the expressiveness of spider diagrams [28].

Spider diagrams have been used for specifying safety critical systems [3], visualizing clusters which contain concepts from multiple ontologies [14] and for allowing files to be viewed in multiple directories [4] as well as having many other applications. Furthermore, they form the basis of constraint diagrams, introduced by Kent [18], which are designed for object oriented specification [15, 20]. Kent's aim, when he introduced constraint diagrams, was to provide a user friendly, formal notation that is well suited to those who like to use diagrams for modelling but shy away from the textual languages that are currently on offer for formal specification. The hope is that, by providing sufficiently expressive formal diagrammatic notations, the use of formal methods will be encouraged leading to improved software design and, as a result, more reliable software will be built.

In order for Kent's vision to be realized, the syntax and semantics of constraint diagrams must be formalized (done in [6]). Furthermore, to enable software engineers to reason about their models, it is essential that sound and, where possible, complete sets of reasoning rules are specified. Two sound and complete systems have been developed for fragments of the langauge [26, 27] and sound rules have been defined for the full notation [5]. It is desirable to develop automated theorem provers for the constraint diagram language, thus providing practical support for software engineers to reason about their models. For example, it may be necessary to prove that two specifications are equivalent or that the post-condition of one operation implies the precondition of another.

The diagrams we have seen so far are all instances of *unitary diagrams*. Unitary diagrams can be joined using logical connectives such as $\wedge$ and $\vee$. Many diagrammatic systems have incorporated logical connectives, for example [5, 16, 25, 26], most of which are sound and complete. Unitary diagrams represent information more concisely than compound diagrams. The ability to express statements concisely could enhance the usability of a language as well as be important for theoretical reasons (discussed in section 2). An interesting question arises: is the unitary spider diagram system as expressive as the full system? The answer is, perhaps unfortunately, no. There are numerous examples of simple statements, as well as complex statements, that cannot be made by any unitary spider diagram that can be made by a compound diagram.

In this paper, we explore deficiencies in the expressive power of unitary spider diagrams. We generalize the syntax of spider diagrams, increasing the expressiveness of the unitary system, overcoming some of these deficiencies. These generalizations give rise to a more flexible system because there are more ways of expressing a given piece of information. As a consequence, it may be that there is a more natural mapping from a statement a user wishes to make to a diagram expressing that statement. We give our motivation for increasing the expressiveness of unitary spider diagrams in section 2. In section 3 we give a brief overview of the syntax and semantics of existing spider diagram systems. In sections 4 to 6 we present our generalizations. We give expressiveness results for the non-generalized and generalized unitary fragments in section 7, where expressiveness limitations of the generalized system are discussed and further generalizations are proposed.

2 Motivational Discussion

There are theoretical reasons for increasing the expressiveness of the unitary system. Firstly, there is interest in automatically generating proofs in spider diagram systems [11, 22]. Approaches have been developed that produce shortest proofs [8, 9]. These approaches use a heuristic function to guide the theorem prover towards good reasoning steps, reducing the amount of backtracking required and, hence, smaller search trees are produced. The heuristic function gives a numerical score that provides a lower bound on the length of a shortest proof from the premise diagram to the conclusion diagram.

Defining an accurate heuristic function for the compound system is challenging, partly due to the tree structure of the diagrams. Even if an accurate heuristic function can be defined, the size of the search tree can still explode due to an abundance of highly applicable reasoning rules causing the nodes of the search trees to have large out-degrees. One problematic rule is idempotency which can be applied to any diagram: from d_1 we can deduce $d_1 \vee d_1$, for example. Given the potentially large number of sub-diagrams in a compound diagram, it is easy to see that the idempotency rule can, sometimes, be applied in many ways. This high applicability makes it hard for the size of the search tree to be controlled. Furthermore, some other highly applicable rules are non-deterministic, such as the information weakening rule 'from d_1 we can deduce $d_1 \vee d_2$' and the

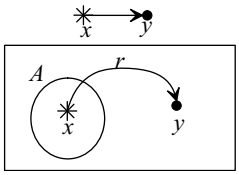

Figure 3. A constraint diagram.

information preserving absorption rule 'from d_1 we can deduce $d_1 \wedge (d_1 \vee d_2)$' (in both cases d_2 is any diagram). This non-determinism makes implementing theorem provers that operate with a complete set of rules for the compound system a challenging task. By contrast, the rules that exist for unitary diagrams are all deterministic.

The effectiveness of the heuristic function can be analyzed by comparing the sizes of the search trees generated by the theorem prover to those generated by conducting a breadth first search. Analysis has shown that restricting the theorem prover to the unitary fragment (without the generalizations presented in this paper) results in a much larger percentage reduction in the size of the search tree when compared with the compound case [8, 9]. We conjecture that similar results can be observed when comparing the more expressive unitary system suggested in this paper with the compound system; if this is the case then increasing the expressiveness of the unitary system will result in more theorems being provable in a reasonable amount of time. Given the inherent difficulty of developing efficient automated theorem provers, any generalizations that may result in more effective theorem proving techniques being developed should be thoroughly investigated.

We note that some people may prefer to use spider diagrams that do not include our generalizations. However, it is simple to translate proofs involving generalized spider diagrams into proofs that involve (non-generalized) spider diagrams. Thus, if we can develop more efficient theorem proving techniques for the generalized system then we can pass on these efficiency savings to the non-generalized system.

Secondly, we wish to be able to automate the drawing of diagrams (this is essential if we are to present automatically generated proofs to users in a diagrammatic form). Considerable research has been conducted into the generation of Euler diagrams [1, 2, 7, 19, 24, 32] and spider diagrams [21]. It can be time consuming to automatically draw visually pleasing Euler diagrams; see [10] for related work. It is preferable to automatically draw unitary diagrams instead of compound diagrams.

Finally, the constraint diagram language [6] extends the spider diagram language. The diagram in figure 3 is a constraint diagram and expresses

$$\forall x \in A\, \exists y \in U - A\big(\, r(x,y) \wedge \forall z\, (r(x,z) \Rightarrow y = z)\big).$$

In the constraint diagram language it can be difficult, maybe impossible, to make some first order statements involving disjunction inside the scope of a universal quantifier; see [29] for a discussion of this issue. It may well be that if the generalizations we propose are incorporated into the constraint diagram language then the expressiveness of the whole system is increased, not just that of the unitary fragment. Indeed, the task of finding efficient theorem proving algorithms for constraint diagrams is daunting because these diagrams are highly expressive. The better we understand how to control the search for proofs in spider diagram systems, the more tractable this task becomes for constraint diagrams. The work presented in this paper is an essential basis for the thorough exploration of theorem proving techniques for spider diagram systems, which will allow us to find highly efficient theorem proving algorithms.

3 Informal Syntax and Semantics

Various systems of spider diagrams have been developed, for example [16, 28], and in this section we give an informal overview of their syntax and semantics.

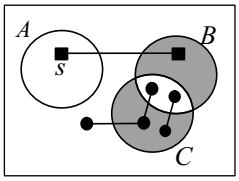

Figure 4. A unitary spider diagram.

A **contour** is a simple closed plane curve. Each contour is labelled. A **boundary rectangle** properly contains all contours. The boundary rectangle is not a contour and is not labelled. A **basic region** is a maximal, bounded set of points in the plane enclosed by a contour or the boundary rectangle. A **region** is defined recursively as follows: any basic region is a region; if r_1 and r_2 are regions then the union, intersection and difference of r_1 and r_2 are regions provided these are non-empty. A **zone** is a region having no other region contained within it. A region is **shaded** if each of its component zones is shaded. The diagram in figure 4 contains three contours, labelled A, B and C. There is one zone inside A and two zones inside B. In total there are five zones (including the zone which is outside all three contours) of which two are shaded.

A **spider** is a tree with nodes (called **feet**) placed in zones. The connecting edges (called **legs**) are straight lines. The feet of a spider are either all square (a **constant spider**) or all round (an **existential spider**). Each constant spider is labelled. A spider **touches** a zone if one of its feet is placed in that zone. A spider can touch a zone at most once. In

figure 4 there are three spiders. The constant spider labelled s has a two zone habitat and, therefore, touches two zones.

A **unitary spider diagram** is a single boundary rectangle together with a finite collection of contours, shading and spiders. No two contours (constant spiders) in the same unitary diagram can have the same contour label (constant spider label). Unitary spider diagrams can be joined together using the logical connectives $\wedge$ and $\vee$ to form **compound diagrams**.

We now describe, informally, the semantics of spider diagrams, see [28, 30] for formal semantics. Contours represent sets. In our discussion, we will identify contour labels with the sets they represent. The diagram in figure 4 asserts that A and B are disjoint, for example, because the contours labelled A and B do not overlap. Zones and regions in a unitary diagram d also represent sets. The zone inside B but outside A and C in figure 4 represents the set $B \cap \big(U - (A \cup C)\big)$ where U is the universal set. A region represents the union of the sets represented by its constituent zones.

Spiders denote elements in the sets represented by their habitats and the spider type affects the precise meaning. Constant spiders denote specific individuals. As with contour labels, in our informal discussion we will identify constant spider labels with the individuals that they represent. The diagram in figure 4 expresses that $s \in A$ or $s \in B - C$. Existential spiders denote the existence of elements. The two existential spiders in figure 4 denote the existence of two distinct elements, one in the set C, the other in $C \cup \big(U - (A \cup B \cup C)\big)$. Since distinct spiders in a unitary diagram denote distinct elements it follows that spiders allow us to place lower bounds on set cardinality.

Shading allows us to place upper bounds on set cardinality. In the set represented by a shaded region, all of the elements are denoted by spiders. For example, the shaded zone inside C in figure 4 represents a set with at most two elements because it is touched by two existential spiders.

If $D = D_1 \vee D_2$ ($D = D_1 \wedge D_2$) is a compound diagram then the semantics of D are the disjunction (conjunction) of the semantics of D_1 and D_2. Unlike unitary diagrams, not all compound diagrams are satisfiable.

Spider diagrams are equivalent in expressive power to monadic first order logic with equality [28, 30]. The unitary fragment is less expressive than the full system. Suppose a statement, S, is made by compound diagram D. If there is no unitary diagram semantically equivalent to D then we say that S **cannot be expressed** by a unitary diagram.

4 Generalizing Constant Spider Labelling

An example of a statement that cannot be expressed by any unitary diagram is s is in A or t is outside A where s and t are specific individuals. This can be expressed by the compound diagram formed by taking the disjunction of d_1 and d_2 in figure 5. If, instead of labelling each constant

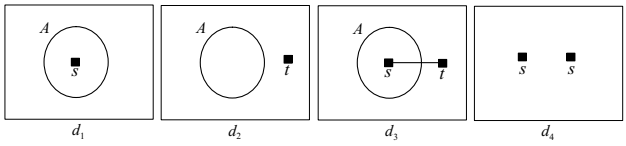

Figure 5. Labelling constant spider feet.

spider, we label each constant spider foot (the labels on distinct feet may be the same or different and each label may be used on multiple spiders), then the unitary diagram d_3 in figure 5 expresses s is in A or t is outside A, because spider legs represent disjunction, and is more concise than $d_1 \vee d_2$.

An interesting point is that, with these generalizations, contradictions can be made by unitary diagrams. For example, d_4 in figure 5 asserts that $s \neq s$ and is, therefore, unsatisfiable. The semantics of unitary diagrams containing these generalized constant spiders are more subtle when many constant spiders are present. Since distinct spiders denote distinct objects, the diagram d_1 in figure 6 asserts that there are two distinct individuals, one is either s in A or t outside A and the other is u. Note that d_1 does not imply that $s \neq u$, $s \neq t$ nor $t \neq u$. Without these generalizations, any unitary diagram that makes an explicit statement about s, t and u would assert that s, t and u are pairwise distinct. The diagrams d_1 and non-generalized $d_2 \vee d_3$ are semantically equivalent but d_1 is more concise.

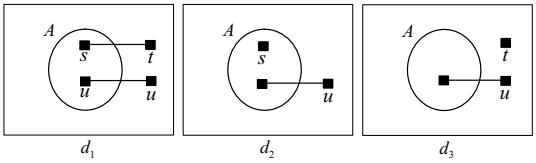

Figure 6. Interacting spiders.

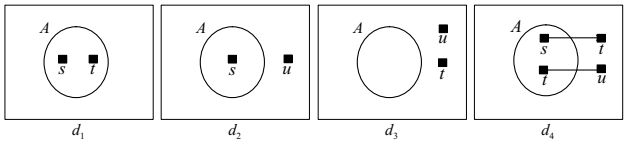

Figure 7. Multiple label use.

Suppose that we wish to assert that there are two distinct individuals, one is either s in A or t outside A and the other is either t in A or u outside A. Without our generalizations, this can only be expressed (implicitly) by a compound diagram, such as $d_1 \vee d_2 \vee d_3$ in figure 7. There is not a direct mapping from the statement to the diagram since $d_1 \vee d_2 \vee d_3$ explicitly expresses

1. there are two distinct individuals, s and t, both in A or
2. there are two distinct individuals, one of which, s, is in A, the other, u, is outside A or
3. there are two distinct individuals, t and u, both outside A.

This example shows that in order to make our required statement using the non-generalized syntax, we must first perform some reasoning to convert our statement into an explicitly representable form and then draw a diagram(s). Relaxing the constraint that each constant spider label occurs at most once in any unitary diagram allows us to express our required statement naturally and concisely, using d_4.

5 Multiple Typed Spiders

A restriction that, up until now, has been placed on spiders is that legs can only join feet of the same type. We generalize spiders so that feet of different types can be connected by legs. In figure 8, the spider in d_1 asserts that either

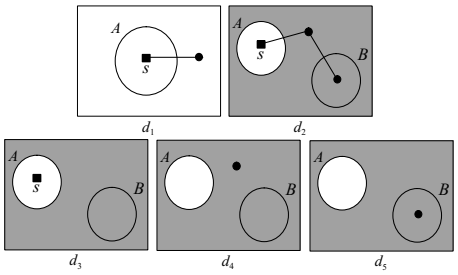

Figure 8. Multiple typed spiders.

s is in A or there is an element outside A. The diagram d_2 expresses that A and B are disjoint and either s is in A or there is an element outside A and, in addition, there are no other elements are outside A and d_2 is semantically equivalent to non-generalized $d_3 \vee d_4 \vee d_5$. This generalization has an analogy in Peirce's system where he allows $\otimes$-sequences to be joined to o-sequences [23].

6 Generalizing the Placing of Spider's Feet

In all previous spider diagram systems, spiders are permitted to touch any given zone at most once. Indeed, allowing spiders to touch a zone, z, more than once does not necessarily provide any more information than a single foot placed in z. For example, the diagram d_1 in figure 9 expresses that there is an element in A or in A, which is semantically equivalent to d_2. The semantics of generalized constant spiders with multiple feet placed in a zone are more interesting. The diagram d_3 asserts that A contains a single

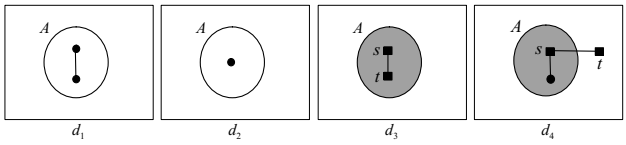

Figure 9. Touching zones many times.

element which is the individual s or the individual t; more formally $A = \{s\} \vee A = \{t\}$. It is not the case that d_3 expresses $s \neq t$. The diagram d_4 expresses t is outside A or s is in A or there is an unspecified element in A and, in addition, nothing else is in A.

7 Expressiveness

All of the generalizations that we have suggested do not change the basic building blocks of the spider diagram language, provided we only use round spider feet (there is no theoretical reason why constant spider feet must be square). The basic building blocks of spiders are feet, constant spider labels and legs. We have removed restrictions placed on how these building blocks can be joined together to make statements. It is easy to prove that the generalizations we have proposed increase the expressiveness of the unitary fragment. It is more interesting to consider how much we have increased expressiveness with our generalizations. Identifying the expressive power of both the non-generalized and the generalized unitary fragments will establish this. We identify fragments of Monadic First Order Logic with equality (MFOLe) equivalent to each of the unitary fragments. We use contour labels as monadic predicate symbols and constant spider labels as constant symbols in MFOLe.

In unitary diagrams, we say which set an element belongs to by placing a spider in the appropriate region of a diagram. In MFOLe, we describe the set that an element belong to using a formula. For example, the statement $\exists x A(x)$ expresses that there is an element in the set A, which is also expressed by d_2 in figure 9 because there is an existential spider placed inside A.

Definition 7.1 *Let x be a variable. A **placement formula** in x is defined inductively. For all monadic predicate symbols, A, the atomic formula $A(x)$ is a placement formula in x. If P and Q are placement formulas in x then so are $(P \wedge Q)$, $(P \vee Q)$ and $\neg P$. For all constant symbols c, $(x = c \wedge P)$ is an **extended placement formula** in x given c.*

In figure 7, non-generalized d_1 is equivalent to

$$\exists x_1 \, \exists x_2 \, \big(x_1 = s \wedge A(x_1) \wedge x_2 = t \wedge A(x_2) \wedge x_1 \neq x_2\big).$$

The structure of the MFOLe sentence above gives a clear indication of the type of MFOLe sentences that can be expressed by non-generalized unitary diagrams. However, in

the particular example above, d_1 does not have any shaded zones. Shading brings with it implicit universal quantification and any fragment of MFOLe that corresponds to the non-generalized unitary system must include the syntax required to place upper bounds on set cardinality. For example, in figure 8, non-generalized d_4 is equivalent to

$$\exists x_1\big(\neg A(x_1) \wedge \neg B(x_1) \wedge \forall x_2\big((A(x_2) \wedge \neg B(x_2)) \vee x_2 = x_1\big)\big).$$

The shading in d_4 corresponds to the universally quantified sub-formula in the equivalent MFOLe sentence above. In general, any element is either in a set represented by a non-shaded zone or is represented by a spider.

Definition 7.2 *Let V be a finite set of variables and let x be a variable not in V. Let P be a placement formula in x. A **bounding formula** in x given V is a formula of the form*

$$\forall x\big(P \vee \bigvee_{y \in V} x = y\big)$$

where an empty disjunction is taken as $\bot$.

We are now in a position to define MFOLe sentences that are expressible by our non-generalized unitary system using placement formulas and bounding formulas as our basis. We recall that non-generalized unitary diagrams are all satisfiable but *expressible sentences*, defined below, can be unsatisfiable. However, all satisfiable expressible sentences are equivalent to some non-generalized unitary diagram.

Definition 7.3 *An **expressible sentence** is defined as follows.*

1. *The true symbol $\top$, is an expressible sentence.*
2. *Any bounding formula with no free variables (that is, the disjunction over V is empty) is an expressible sentence.*
3. *Any sentence of the form*

$$\exists x_1 ... \exists x_n \Big(\bigwedge_{1 \le i \le n} P_i \wedge \bigwedge_{1 \le i \le n-1} x_i \neq x_{i+1} \wedge Q \Big)$$

is an expressible sentence provided

   (a) *each P_i is either a placement formula in x_i or an extended placement formula in x_i given some constant c,*

   (b) *for each P_i and P_j if P_i and P_j are extended placement formulas given constants c_i and c_j respectively then $c_i = c_j$ implies $P_i = P_j$ and*

   (c) *Q is either $\top$ or a bounding formula given $V = \{x_1, ..., x_n\}$ in some variable, x, not in V.*

For example, the MFOLe sentence below is expressible, being equivalent to non-generalized d_1 in figure 10:

$$\exists x_1 \exists x_2\big(x_1 = s \wedge A(x_1) \wedge B(x_2) \wedge x_1 \neq x_2 \wedge \forall x_3(\neg B(x_3) \vee x_3 = x_1 \vee x_3 = x_2)\big).$$

Theorem 7.1 *The non-generalized unitary fragment is equivalent in expressive power to the class of satisfiable expressible sentences.*

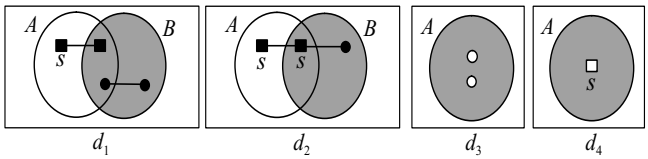

Figure 10. Expressing sentences.

Our attention now turns to the generalized case. All sentences that are expressible by non-generalized unitary diagrams are obviously expressible by generalized unitary diagrams. The generalizations we have proposed allow more disjunctive statements to be made by unitary diagrams. For example, in figure 10, generalized d_2 is equivalent to the *generalized expressible sentence*

$$\exists x_1\big(\big((x_1 = s \wedge A(x_1)) \vee (B(x_1) \wedge \neg A(x_1))\big) \wedge \forall x_2\big(\neg B(x_2) \vee x_2 = x_1\big)\big).$$

Definition 7.4 *A **generalized expressible sentence** is defined as follows.*

1. *Any expressible sentence is a generalized expressible sentence.*
2. *Any sentence of the form*

$$\exists x_1 ... \exists x_n \Big(\bigwedge_{1 \le i \le n} R_i \wedge \bigwedge_{1 \le i \le n-1} x_i \neq x_{i+1} \wedge Q \Big)$$

is a generalized expressible sentence provided

   (a) *R_i is any finite disjunction of placement formulas in x_i or extended placement formulas in x_i given some constant c,*

   (b) *Q is either $\top$ or a bounding formula given $V = \{x_1, ..., x_n\}$ in some variable, x, not in V.*

Theorem 7.2 *The generalized unitary fragment is equivalent in expressive power to the class of generalized expressible sentences.*

Theorem 7.3 *The generalized unitary fragment is more expressive than the non-generalized unitary fragment.*

Theorems 7.1 and 7.2 show how our generalizations have increased the expressiveness of the unitary system. Furthermore, they show the type of statements that each unitary system is capable of expressing, thus highlighting expressive limitations. For example, no generalized unitary diagram can express either of the two statements $\forall x\, \neg A(x) \vee \forall x\, \neg B(x)$ and $\forall x\, \neg A(x) \vee \exists x_1 \exists x_2\ x_1 \neq x_2$. Peirce's o-sequences can be imported into spider diagrams to make the first statement but there is no obvious generalization that allows us to make the latter.

A further limitation is that no generalized unitary diagram can express arbitrary finite lower and upper bounds on any given set *without* also specifying further information, exemplified by the example $0 \leq |A| \leq 2$. *Schrödinger spiders*, introduced in [12], have round unfilled feet and express that an element might be present. Including such spiders in the language allows us to express $0 \leq |A| \leq 2$, shown by d_3 in figure 10. We can also introduce a new type of spider, called a **Schrödinger constant spider**, in order to express that an individual might be in a particular set, see d_4 in figure 10 which expresses $A \subseteq \{s\}$. Schrödinger spider feet, of either type, can also be joined to other spiders following all of the generalizations presented in the previous sections.

8 Conclusions and Open Problems

In this paper we have explored generalizations of spider diagrams that increase the expressiveness of the unitary fragment. By allowing constant spiders' feet to be labelled, for example, we have provided a natural way of making some simple statements using unitary diagrams. Indeed, all of the extensions we have proposed enhance the spider diagram system by making it more flexible and notationally efficient. We believe that our generalizations will make it easier for people to make certain statements. This is suggested by the fact that some statements can be explicitly made by generalized unitary diagrams which can only be implicitly made by the non-generalized (compound) system. However, the increased complexity of the generalized notation may mean that some people find certain generalized unitary diagrams harder to interpret than semantically equivalent non-generalized compound diagrams. It is straightforward to convert generalized diagrams into non-generalized diagrams. With appropriate tool support, such a conversion can be easily automated. It will be interesting to establish whether generalized spider diagrams are more effective, in terms of promoting human task efficiency, than non-generalized spider diagrams.

A particular challenge faced when attempting to increase the expressiveness of the unitary fragment further is how best to express disjunctive information that involves shading. Incorporating Peirce's o-sequences, as discussed above, will yield some increase in expressiveness but will not entirely solve the problem. Shading brings with it (implicit) universal quantification. To maximize expressiveness we need to interpret shading and spiders (which have existential import) in arbitrary orders which can make defining semantics difficult. It is likely that a dependence analysis would be required, as in the constraint diagram language [6]. We conjecture that such an extension of the unitary fragment will be as expressive as the full system.

The development of reasoning rules is necessary before we can investigate the effect of our generalizations on the ability of our theorem prover to find proofs. It will be interesting, and difficult, to provide a complete classification of the proof tasks where our generalizations are beneficial in terms of time taken to find a proof.

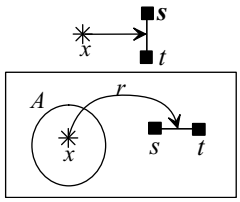

Figure 11. A generalized constraint diagram.

The constraint diagram language can be extended by incorporating our generalizations. The generalized constraint diagram in figure 11 expresses the same as the first order logic sentence

$$\forall x \in A(r(x,s) \vee r(x,t)) \wedge \forall y\big(r(x,y) \Rightarrow (y = s \vee y = t)\big).$$

The *reading tree* above the diagram informs us that we are to read the asterisk (a universal spider) before the constant spider. It is not obvious whether *any* non-generalized constraint diagram can make such a statement. Whilst the exact expressive of constraint diagrams is unknown, we believe that our generalizations will lead to an increase in expressiveness of the compound constraint diagram system as well as the unitary fragment.

Acknowledgements Gem Stapleton is supported a by Leverhulme Trust Early Career Fellowship. Thanks to John Taylor for his comments on earlier drafts of this paper.

References

[1] S. Chow and F. Ruskey. Drawing area-proportional Venn and Euler diagrams. In *Proceedings of Graph Drawing 2003, Perugia, Italy*, volume 2912 of *LNCS*, pages 466–477. Springer-Verlag, September 2003.

[2] S. Chow and F. Ruskey. Towards a general solution to drawing area-proportional Euler diagrams. In *Proceedings of Euler Diagrams*, volume 134 of *ENTCS*, pages 3–18, 2005.

[3] R. 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.

[4] 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.

[5] 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.

[6] A. Fish, J. Flower, and J. Howse. The semantics of augmented constraint diagrams. *Journal of Visual Languages and Computing*, 16:541–573, 2005.

[7] J. Flower and J. Howse. Generating Euler diagrams. In *Proceedings of 2nd International Conference on the Theory and Application of Diagrams*, pages 61–75, Georgia, USA, April 2002. Springer-Verlag.

[8] J. Flower, J. Masthoff, and G. Stapleton. Generating proofs with spider diagrams using heuristics. In *Proceedings of Distributed Multimedia Systems, International Workshop on Visual Languages and Computing*, pages 279–285. Knowledge Systems Institute, 2004.

[9] 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.

[10] J. Flower, P. Rodgers, and P. Mutton. Layout metrics for Euler diagrams. In *7th International Conference on Information Visualisation*, pages 272–280. IEEE Computer Society Press, 2003.

[11] J. Flower and G. Stapleton. Automated theorem proving with spider diagrams. In *Proceedings of Computing: The Australasian Theory Symposium*, volume 91 of *ENTCS*, pages 116–132, Dunedin, New Zealand, January 2004. Science Direct.

[12] J. Gil, J. Howse, and S. Kent. Formalising spider diagrams. In *Proceedings of IEEE Symposium on Visual Languages (VL99), Tokyo*, pages 130–137. IEEE Computer Society Press, September 1999.

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

[14] 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.

[15] J. Howse and S. Schuman. Precise visual modelling. *Journal of Software and Systems Modeling*, 4:310–325, 2005.

[16] J. Howse, G. Stapleton, and J. Taylor. Spider diagrams. *LMS Journal of Computation and Mathematics*, 8:145–194, 2005.

[17] M. Jamnik. *Mathematical Reasoning with Diagrams*. CSLI, 2001.

[18] S. Kent. Constraint diagrams: Visualizing invariants in object oriented modelling. In *Proceedings of OOPSLA97*, pages 327–341. ACM Press, October 1997.

[19] H. Kestler, A. Muller, H. Liu, D. Kane, B. Zeeberg, and J. Weinstein. Euler diagrams for visualizing annotated gene expression data. In *Proceedings of Euler Diagrams 2005*, Paris, September 2005.

[20] 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.

[21] P. Mutton, P. Rodgers, and J. Flower. Drawing graphs in Euler diagrams. In *Proceedings of 3rd International Conference on the Theory and Application of Diagrams*, volume 2980 of *LNAI*, pages 66–81, Cambridge, UK, March. Springer.

[22] O. Patrascoiu, S. Thompson, and P. Rodgers. Tableaux for diagrammatic reasoning. In *Proceedings of the Eleventh International Conference on Distributed Multimedia Systems, International Workshop on Visual Languages and Computing*, pages 279–286. Knowledge Systems Institute, September 2005.

[23] C. Peirce. *Collected Papers*, volume 4. Harvard University Press, 1933.

[24] P. Rodgers, P. Mutton, and J. Flower. Dynamic Euler diagram drawing. In *Visual Languages and Human Centric Computing, Rome, Italy*, pages 147–156. IEEE Computer Society Press, September 2004.

[25] S.-J. Shin. *The Logical Status of Diagrams*. Cambridge University Press, 1994.

[26] G. Stapleton, J. Howse, and J. Taylor. A constraint diagram reasoning system. In *Proceedings of International Conference on Visual Languages and Computing*, pages 263–270. Knowledge Systems Insitute, 2003.

[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. Howse, J. Taylor, and S. Thompson. The expressiveness of spider diagrams augmented with constants. In *Visual Languages and Human Centric Computing, Rome, Italy*, pages 91–98. IEEE Computer Society Press, September 2004.

[29] G. Stapleton, S. Thompson, A. Fish, J. Howse, and J. Taylor. A new language for the visualisation of logic and reasoning. In *11th International Conference on Distributed Multimedia Systems, International Workshop on Visual Languages and Computing*. Knowledge Systems Institute, September 2005.

[30] 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.

[31] N. Swoboda and G. Allwein. Using DAG transformations to verify Euler/Venn homogeneous and Euler/Venn FOL heterogeneous rules of interence. In *Proceedings of GT-VMT*, ENTCS. Elsevier Science, 2002.

[32] A. Verroust and M.-L. Viaud. Ensuring the drawability of Euler diagrams for up to eight sets. In *Proceedings of 3rd International Conference on the Theory and Application of Diagrams*, volume 2980 of *LNAI*, pages 128–141, Cambridge, UK, 2004. Springer.

[33] D. Winterstein, A. Bundy, and C. Gurr. Dr Doodle: A diagrammatic theorem prover. In *Proceedings of The International Joint Conference on Automated Reasoning*, LNCS. Springer-Verlag, 2004.