Reasoning with Constraint Diagrams

Gem Stapleton

The Visual Modelling Group
University of Brighton, Brighton, UK
www.cmis.brighton.ac.uk/research/vmg
g.e.stapleton@brighton.ac.uk
Technical Report VMG.04.01

Abstract

Constraint diagrams are designed for the formal specification of software systems. However, their applications are broader than this since constraint diagrams are a logic that can be used in any formal setting. This document summarizes the main results presented in my PhD thesis, the focus of which is on a fragment of the constraint diagram language, called spider diagrams, and constraint diagrams themselves. In the thesis, sound and complete systems of spider diagrams and constraint diagrams are presented and the expressiveness of the spider diagram language is established.

1 Introduction

The focus of this thesis is the constraint diagram language, introduced by Kent [18] for use by software engineers for formal specification. These diagrams are designed to integrate well with the diagrammatic notations in the Unified Modelling Language (UML) and are an alternative to the UML's textual Object Constraint Language. The constraint diagram in Fig. 1 is an example of a system invariant that we might wish to place on a library lending system. It expresses, amongst other things, that every person can only borrow books that are in the collections of libraries that they have joined.

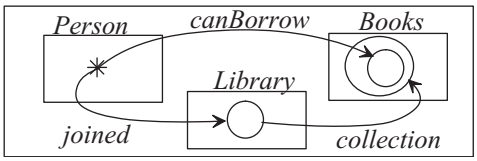

Figure 1: A constraint diagram.

Indeed, one might argue that diagrams cannot be used as a formal tool but only serve as an aid to understanding symbolic (or textual) statements and proofs. In seminal work, Shin established that diagrammatic logics could be given the same status as traditional textual logics by defining a formal, sound and complete diagrammatic reasoning system [22]. Furthermore, Shin proved that her so-called Venn-II system is equivalent in expressive power to monadic first order predicate logic. In the last decade, many more diagrammatic logics have emerged, most of which are sound and complete, for example [2, 9, 10, 13, 14, 15, 17, 20, 23, 24].

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 the only viable option 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. 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. It is also desirable to identify decision procedures for decidable fragments of the language so that we can determine whether software specifications are valid or contradictory. It is important to establish the expressive power of the constraint diagram language (or fragments of it) so that we are aware of its limitations in terms of what can be formally specified. Whilst it is all to easy to identify such tasks, their solution is not necessarily straightforward.

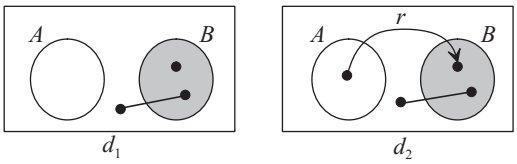

Figure 2: A spider diagram and a constraint diagram.

Much work has been conducted on a fragment of the constraint diagram language called the spider diagram language [8, 12, 13, 14, 15, 20]. Capable of expressing arbitrary finite lower and upper bounds on set cardinality, spider diagrams (and constraint diagrams) are based on Euler diagrams augmented with shading and *spiders*. In Fig. 2, d_1 is a spider diagram and expresses that $A \cap B = \emptyset$, $1 \leq |B| \leq 2$ and $|U - A| \geq 2$ where U is the universal set. The trees in d_1 are examples of spiders. In both spider and constraint diagrams, spiders represent the existence of elements and, in constraint diagrams only, can be used to make universally quantified statements. Unlike spider diagrams, constraint diagrams can express statements involving dyadic predicate symbols by using labelled arrows. The diagram d_2 in Fig. 2 is a constraint diagram and, in addition to the information expressed by the underlying spider diagram, it expresses that there is an element in A that is related to exactly one element, x, under the relation r and $x \in B$. The diagram is semantically equivalent to the first order predicate logic sentence

$$\forall x_1 \neg(A(x_1) \wedge B(x_1)) \wedge \exists x_1 \exists x_2 \exists x_3 \Big(\neg(x_1 = x_2) \wedge \neg(x_1 = x_3) \wedge \neg(x_2 = x_3) \wedge$$
$$A(x_1) \wedge B(x_2) \wedge \neg A(x_3) \wedge \forall x_4 \big(B(x_4) \Rightarrow (x_4 = x_2 \vee x_4 = x_3)\big) \wedge r(x_1, x_2) \wedge \forall x_4 \big(r(x_1, x_4) \Rightarrow x_4 = x_2\big) \Big).$$

Prior to the work in this thesis, various sound and complete systems of spider diagrams have been developed, all of which are based on Venn diagrams[1] and restricted to making statements in a conjunctive normal form. The first contribution that this thesis makes is to formalize a spider diagram logic based on the more user friendly Euler diagrams and the restriction to CNF is removed. Reasoning rules are defined and the system is shown to be sound, complete and decidable. This work is elaborated in section 2. Secondly, the spider diagram logic is shown to be equivalent in expressive power to monadic first order logic with equality. Hence, spider diagrams are more expressive than Shin's Venn-II system. The expressiveness work is detailed in section 3. The third contribution, discussed in section 4, is the formalization of a relatively highly expressive fragment of the constraint diagram language, for which sound and complete reasoning rules are provided. This is the first constraint diagram reasoning system. Whilst the basic completeness proof strategy is reasonably straightforward the details are difficult. A decision procedure can be extracted from the proof, showing that the system is decidable.

2 Spider Diagrams

In this section, we give a brief overview of the spider diagram system developed in the thesis. We specify the formal syntax and semantics (this will be useful to us in section 3) and include one of the reasoning rules that is defined for the system. The work on constraint diagrams discussed in section 4 directly extends the spider diagram system and more reasoning rules will be illustrated there.

2.1 Syntax

The spider diagram d_1 in Fig. 3 has two *contours*: these are the circles labelled A and B; in general contours are simple closed curves. This diagram also has four *zones*: these are the minimal regions in d_1. For example, one zone is inside the contour A and outside the contour B. The shaded zone in d_1 is the *habitat* of one spider (the habitat of a spider is the set of zones that the spider is placed in)and is *touched* by two spiders (a spider touches a zone provided that zone is in its habitat).

The contour labels used in our diagrams are chosen from a countably infinite set, $\mathcal{L}$. A zone can be described by the labels of the contours that it is inside and the contours that it is outside. Formally, we define a **zone** to be a pair of finite, disjoint sets of contour labels. The zone (a, b) is included in a but not included in b. A **region** is a set of zones.

[1] In a Venn diagram, all of the intersections between the sets referenced in the diagram must be explicitly represented.

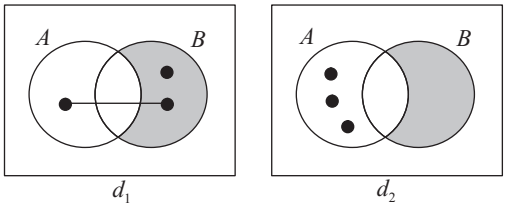

Figure 3: Two spider diagrams

Definition 1. *An **abstract unitary spider diagram**, d, (with labels in $\mathcal{L}$) is a tuple $\langle L, Z, Z^*, SI\rangle$ whose components are defined as follows.*

1. *$L = L(d) \subset \mathcal{L}$ is a finite set of contour labels.*
2. *$Z = Z(d) \subseteq \{(a, L-a) : a \subseteq L\}$ is a set of zones such that*
   (i) *for each label $l \in L$ there is a zone $(a, L-a) \in Z(d)$ such that $l \in a$ and*
   (ii) *the zone $(\emptyset, L)$ is in $Z(d)$.*
3. *$Z^* = Z^*(d) \subseteq Z$ is a set of **shaded zones**.*
4. *$SI = SI(d) \subset \mathbb{Z}^+ \times (\mathbb{P}Z - \{\emptyset\})$ is a finite set of **spider identifiers** such that*

$$\forall (n_1, r_1), (n_2, r_2) \in SI \bullet r_1 = r_2 \Rightarrow n_1 = n_2$$

*If $(n, r) \in SI$ we say there are n **spiders** with **habitat** r.*

The symbol $\perp$, is also a unitary diagram. If d_1 and d_2 are spider diagrams then so are $(d_1 \sqcup d_2)$ ('d_1 or d_2') and $(d_1 \sqcap d_2)$ ('d_1 and d_2').

Every contour in a drawn diagram contains at least one zone, captured by condition 2 (i). In any drawn diagram, the zone inside the rectangle but outside all the contours is present, captured by condition 2 (ii). The definition of a spider diagram given here is abstract but the thesis also contains the formal syntax of drawn (or *concrete*) spider diagrams and mappings are given between these two levels of syntax. The motivation for using two levels of syntax can be found in [11] and some previous spider diagram systems also follow the two level approach. The abstract syntax given here is more elegant than previous formalizations. Consequently, many of the definitions given in the thesis which use the abstract syntax are more succinct than in previous systems.

Example 1. The drawn diagram d_1 in Fig. 3 corresponds to the abstract diagram with

1. contour labels $\{A, B\}$,
2. zones $\{(\emptyset, \{A, B\}), (\{A\}, \{B\}), (\{B\}, \{A\})(\{A, B\}, \emptyset)\}$,
3. shaded zones $\{(\{B\}, \{A\})\}$ and
4. spider identifiers $\{(1, \{(\{B\}, \{A\})\}), (1, \{(\{A\}, \{B\}), (\{B\}, \{A\})\})\}$.

Spiders represent the existence of elements and regions represent sets – thus we need to know how many elements we have represented in each region. Whilst our definition of a spider diagram uses spider identifiers, it is useful (for our semantic interpretation) to define the set of spiders in a diagram. We define, for unitary diagram d, the set of spiders in d to be

$$S(d) = \{s_i(r) : \exists (n, r) \in SI(d) 1 \leq i \leq n\}.$$

The spider $s_i(r)$ has **habitat** r. The set of spiders inhabiting region r in d is denoted by $S(r, d)$ where

$$S(r, d) = \{s_i(r') \in S(d) : r' \subseteq r\}.$$

The number of spiders touching r in d is denoted by $T(r, d)$, where

$$T(r, d) = \{s_i(r') \in S(d) : r' \cap r \neq \emptyset\}.$$

In d_1, Fig. 3, the zone $(\{B\}, \{A\})$ is inhabited by one spider and touched by two spiders.

2.2 Semantics

Regions in spider diagrams represent sets. We can express lower and, in the case of shaded regions, upper bounds on the cardinalities of the sets we are representing as follows. If region r is inhabited by n spiders in diagram d then d expresses that the set represented by r contains at least n elements. If r is shaded and touched by m spiders in d then d expresses that the set represented by r contains at most m elements. Thus, if d has a shaded, untouched region, r, then d expresses that r represents the empty set. Missing zones (zones that could be in the diagram given the label set, but are not present) also represent the empty set. To formalize the semantics we shall map contour labels, zones and regions to subsets of some universal set. We assume that no contour label is a zone or region and that no zone is a region. We define $\mathcal{Z}$ and $\mathcal{R}$ to be the sets of all zones and regions respectively.

Definition 2. *An **interpretation of contour labels, zones and regions**, or simply an **interpretation**, is a pair $m = (U, \Psi)$ where U is a set and $\Psi\colon \mathcal{L} \cup \mathcal{Z} \cup \mathcal{R} \to \mathbb{P}U$ is a function mapping contour labels, zones and regions to subsets of U such that the images of the zones and regions are completely determined by the images of the contour labels as follows:*

1. *for each zone (a, b), $\Psi(a, b) = \bigcap_{l \in a} \Psi(l) \cap \bigcap_{l \in b} \overline{\Psi(l)}$ where $\overline{\Psi(l)} = U - \Psi(l)$ and we define $\bigcap_{l \in \emptyset} \Psi(l) = U = \bigcap_{l \in \emptyset} \overline{\Psi(l)}$ and*
2. *for each region r, $\Psi(r) = \bigcup_{z \in r} \Psi(z)$ and we define $\Psi(\emptyset) = \bigcup_{z \in \emptyset} \Psi(z) = \emptyset$.*

We introduce a *semantics predicate* which identifies whether a diagram expresses a true statement, with respect to an interpretation.

Definition 3. *Let d be a diagram and let $m = (U, \Psi)$ be an interpretation. We define the **semantics predicate**, $P_d(m)$, of d as follows. If $d = \bot$ then $P_d(m) = \bot$. If d ($\neq \bot$) is a unitary diagram then $P_d(m)$ is the conjunction of the following three conditions.*

*(i) **Distinct Spiders Condition.** For each region r in $\mathbb{P}Z(d) - \{\emptyset\}$, $|\Psi(r)| \geq |S(r, d)|$.*

*(ii) **Shading Condition.** For each shaded region r in $\mathbb{P}Z^*(d) - \{\emptyset\}$, $|\Psi(r)| \leq |T(r, d)|$*

*(iii) **The Plane Tiling Condition** All elements are in sets represented by zones: $\bigcup_{z \in Z(d)} \Psi(z) = U$.*

*If $d = d_1 \sqcup d_2$ then $P_d(m) = P_{d_1}(m) \vee P_{d_2}(m)$. If $d = d_1 \sqcap d_2$ then $P_d(m) = P_{d_1}(m) \wedge P_{d_2}(m)$. We say m **satisfies** d, denoted $m \models d$, if and only if $P_d(m)$ is true. If $m \models d$ we say m is a **model** for d.*

Example 2. The interpretation $m = (\{1, 2\}, \Psi)$ partially defined by $\Psi(A) = \{1\}$ and $\Psi(B) = \{2\}$ is a model for d_1 in Fig. 3 but not for d_2.

Every unitary diagram, d, is satisfiable, and this can be shown by constructing a model for d, taking the set of spiders in d as the universal set and mapping contour labels to sets of spiders in an appropriate way.

2.3 Reasoning Rules

Many of the reasoning rules given in the thesis are generalizations of those in [20] for a spider diagram system which is based on Venn diagrams and restricted to CNF. The rules given in this thesis are firstly stated informally, to aid intuition, and then the formal definition using the abstract syntax is provided. Some of the rules have analogies in propositional logic, like distributivity. Unlike some of the 'propositional logic' rules, the truly diagrammatic rules preserve semantic information. Although this is not a requirement, information preserving rules are useful when using semantic tableaux. Since this thesis was completed, a tableaux system for this spider diagram system has been implemented [21]. Indeed, theorem provers have been developed for the spider diagram logic presented here [5, 6, 7]. In this section, we include just one example of a reasoning rule due to space limitations. In total, sixteen rules are defined for the system, giving a complete set.

To motivate the rule illustrated below, we observe that a spider whose habitat is a non-trivial union of regions represents disjunctive information. This can be reflected at the syntactic level by replacing a unitary diagram containing such a spider with a disjunction of unitary diagrams, where we *split* the spider.

Informal Description of the Splitting Spiders Rule. Let d be a unitary diagram with a spider s touching every zone of two disjoint non-empty regions r_1 and r_2. Let d_1 and d_2 be unitary diagrams that are copies of d except that, in d_1, s is replaced by a spider whose habitat is region r_1 and, in d_2, s is replaced by a spider whose habitat is region r_2. Then d can be replaced by the diagram $d_1 \sqcup d_2$.

Example 3. Fig. 4 illustrates an application of the splitting spiders rule. The spider with the three zone habitat in d splits into two spiders, one in d_1, the other in d_2. Intuitively, the element represented by the split spider either belongs to the set $U - (A \cup B)$ or to the set $A \cup B$.

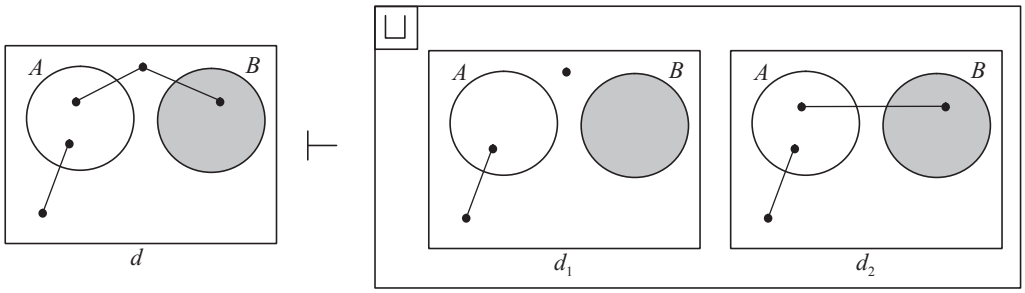

Figure 4: Splitting spiders.

Formal Definition of the Splitting Spiders Rule. Let d be a unitary diagram and let r, r_1 and r_2 be regions of d such that $r = r_1 \cup r_2$ and $r_1 \cap r_2 = \emptyset$. Let $s_n(r)$ be a spider in d (with habitat r). Let d_1 and d_2 be unitary diagrams such that

1. $Z(d) = Z(d_1) = Z(d_2)$,
2. $Z^*(d) = Z^*(d_1) = Z^*(d_2)$,
3. there exist spiders $s_1 \in S(d_1)$ and $s_2 \in S(d_2)$ such that the habitat of s_1 is r_1, the habitat of s_2 is r_2 and

$$S(d) - \{s_n(r)\} = S(d_1) - \{s_1\} = S(d_2) - \{s_2\}.$$

Then d can be replaced by $d_1 \sqcup d_2$ and vice versa.

2.4 Soundness and Completeness

The soundness of the system is demonstrated in the thesis by first proving that each of the sixteen rules are valid (that is, applying the rule maintains or enlarges the class of models of the premise diagram) and then a simple induction argument gives the required result. Some of the validity proofs are straightforward but others are more challenging (like that of the splitting spiders rule). To prove the validity of the splitting spiders rule, we need a result which states that if an interpretation satisfies a unitary diagram d and $d_1 \sqcup d_2$ is a diagram obtained from d by applying the splitting spiders rule, then the distinct spiders condition holds for one of the disjuncts d_1, d_2 or the shading condition holds for the other disjunct.

Lemma 1. *Let d be a unitary diagram and suppose that $d_1 \sqcup d_2$ results on applying the splitting spiders rule to d. Let $m = (U, \Psi)$ be a model for d. Then*

$$\Big(\bigwedge_{r \in R(d_1)} |\Psi(r)| \geq |S(r, d_1)| \vee \bigwedge_{r \in R^*(d_2)} |\Psi(r)| \leq |T(r, d_2)| \Big)$$
$$\wedge \quad \Big(\bigwedge_{r \in R(d_2)} |\Psi(r)| \geq |S(r, d_2)| \vee \bigwedge_{r \in R^*(d_1)} |\Psi(r)| \leq |T(r, d_1)| \Big).$$

Lemma 2. *The splitting spiders rule is valid.*

Sketch of proof Let $m = (U, \Psi)$ be an interpretation. It is easy to show $P_{d_1}(m) \Rightarrow P_d(m)$ and $P_{d_2}(m) \Rightarrow P_d(m)$. Hence $(P_{d_1}(m) \vee P_{d_2}(m)) \Rightarrow P_d(m)$; in other words, $d_1 \sqcup d_2 \vDash d$. To show $P_d(m) \Rightarrow (P_{d_1}(m) \vee P_{d_2}(m))$, show

$$\begin{aligned} P_d(m) \quad \Rightarrow \quad & \bigcup_{z \in Z(d_1)} \Psi(z) = U \wedge \bigcup_{z \in Z(d_2)} \Psi(z) = U \wedge \\ & \Big(\bigwedge_{r \in R(d_1)} |\Psi(r)| \geq |S(r, d_1)| \vee \bigwedge_{r \in R(d_2)} |\Psi(r)| \geq |S(r, d_2)| \Big) \wedge \\ & \Big(\bigwedge_{r \in R^*(d_1)} |\Psi(r)| \leq |T(r, d_1)| \vee \bigwedge_{r \in R^*(d_2)} |\Psi(r)| \leq |T(r, d_2)| \Big). \end{aligned}$$

Then use lemma 1 to give $P_d(m) \Rightarrow (P_{d_1}(m) \vee P_{d_2}(m))$.

All of the rules are shown to be valid using similar proof strategies from which it follows that the system is sound. For space reasons, we omit details of the completeness proof. Very briefly, given spider diagrams d_1 and d_2 such that $d_1 \models d_2$, the strategy is to convert both d_1 and d_2 into a disjunction of unitary diagrams, whilst preserving their semantics, and then reason about these disjunctions. The completeness proof strategy used for the constraint diagram system developed in this thesis extends that for spider diagrams and more details will be provided there.

Theorem 1. ***Soundness and Completeness*** *Let d_1 and d_2 be spider diagrams. Then $d_1 \vdash d_2$ iff $d_1 \models d_2$.*

It can also be shown that the system is decidable.

3 The Expressiveness of Spider Diagrams

Here, we show that the spider diagram logic is equivalent in expressive power to monadic first order logic with equality (MFOLe). To show this equivalence, in one direction, for each diagram we construct a sentence in MFOLe that expresses the same information. We illustrate this conversion by example, omitting the general argument which is given in the thesis. For the significantly more challenging converse we show, for each MFOLe sentence S, there exists a finite set of models for S that can be used to classify all the models for S. Using these classifying models we show that there is a diagram expressing the same information as S.

We shall assume the standard first order predicate logic semantic interpretation of formulae in MFOLe, with the exception of allowing a structure to have an empty domain. An application of this work is modelling object oriented systems, with the universal set containing precisely the objects in existence. There may be no objects, so it is important that we allow empty domains.

3.1 Structures and Interpretations

We wish to identify when a diagram and a sentence express the same information. To aid us formalize this notion, we map interpretations to structures in such a way that information is preserved. Throughout this section we shall assume, without loss of generality, that $\mathcal{L} = \{L_1, L_2, ...\}$ and that we have a countably infinite set of monadic predicate symbols, $\mathcal{P} = \{P_1, P_2, ...\}$. Define $\mathcal{U}$ to be the class of all sets. The sets in $\mathcal{U}$ form the domains of structures in the language MFOLe.

Definition 4. *Define $\mathcal{INT}$ to be the class of all interpretations for spider diagrams. Define also $\mathcal{STR}$ to be the class of structures for the language MFOLe, that is $\mathcal{STR} = \{m : U \in \mathcal{U} \wedge m = \langle U, =^m, P_1^m, P_2^m, ...\rangle\}$, where P_i^m is the interpretation of P_i in the structure m (that is, $P_i^m \subseteq U$) and we always interpret $=$ as the diagonal subset of $U \times U$, denoted $diag(U \times U)$.*

We define a bijection, $h\colon \mathcal{INT} \to \mathcal{STR}$ by $h(U, \Psi) = \langle U, diag(U \times U), \Psi(L_1), \Psi(L_2), ...\rangle$.

Definition 5. *Let d be a spider diagram and S be a MFOLe sentence. We say d and S are **expressively equivalent** if and only if h provides a bijective correspondence between their models, that is*

$$\{h(I) : I \in \mathcal{INT} \wedge I \models d\} = \{m \in \mathcal{STR} : m \models S\}.$$

3.2 Mapping from Diagrams to Sentences

To show that the spider diagram language is not more expressive than MFOLe, we will map diagrams to expressively equivalent sentences. An **α-diagram** is a spider diagram in which all spiders inhabit exactly one zone [20][2].

Theorem 2. *Let d_1 be a spider diagram. There exists a spider diagram, d_2, that is a disjunction of unitary α-diagrams and semantically equivalent to d_1 (that is, d_1 and d_2 have the same models).*

We map each unitary α-diagram to an expressively equivalent sentence in MFOLe. This enables us to map each disjunction of unitary α-diagrams to an expressively equivalent sentence and, by theorem 2, this is sufficient to show that the spider diagram language is not more expressive than the language MFOLe.

[2] α-diagrams play an important role in the completeness proof.

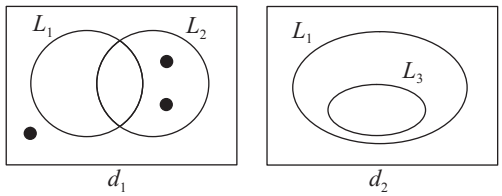

Figure 5: Two α-diagrams: from diagrams to sentences.

Example 4. In diagram d_1, Fig. 5, there are three spiders, one outside both L_1 and L_2, the other two inside L_2 and outside L_1. The diagram d_1 is expressively equivalent to the sentence

$$\exists x_1\,(\neg P_1(x_1) \wedge \neg P_2(x_1)) \wedge \exists x_1 \exists x_2\,(P_2(x_1) \wedge P_2(x_2) \wedge \neg P_1(x_1) \wedge \neg P_1(x_2) \wedge x_1 \neq x_2).$$

In diagram d_2, no elements can be in L_3 and not in L_1, so d_2 is expressively equivalent to sentence

$$\forall x_1\,\neg(P_3(x_1) \wedge \neg P_1(x_1)).$$

We observe that the disjunction of these sentences is expressively equivalent to $d_1 \sqcup d_2$.

In general, for unitary d_1 and d_2, the disjunction of their expressively equivalent sentences is expressively equivalent to $d_1 \sqcup d_2$. It is easy to see how the above example generalizes, hence showing that spider diagrams are at most as expressive as MFOLe.

3.3 Mapping from Sentences to Diagrams

We now consider the significantly more challenging task of constructing a diagram for a sentence, outlining the approach taken. Since every formula is semantically equivalent to a sentence obtained by prefixing the formula with $\forall x_i$ for each free variable x_i (giving its universal closure) we only need to identify a diagram expressively equivalent to each sentence.

Shin's approach for mapping MFOL (without equality) into Venn-II is algorithmic [22], which we now sketch. To find a diagram expressively equivalent to a sentence she first converts the sentence into prenex normal form, say $Q_1x_1...Q_nx_nG$ where G is quantifier free. If Q_n is universal then G is transformed into conjunctive normal form. If Q_n is existential then G is transformed into disjunctive normal form. Quantifier Q_n is then distributed through G and as many formulae are removed from its scope as possible. All n quantifiers are distributed through in this way, thus removing any nesting of quantifiers. A diagram can then be drawn for each of the simple parts of the resulting sentence.

This algorithm does not readily generalize to arbitrary sentences in MFOLe because $=$ is a dyadic predicate symbol which means nesting of quantifiers cannot necessarily be removed. Thus we take a different approach, modelled on what appears in [3], pages 209-210. To establish the existence of a diagram expressively equivalent to a sentence we consider models for that sentence. To illustrate the approach we investigate relationships between models for α-diagrams by considering a particular example.

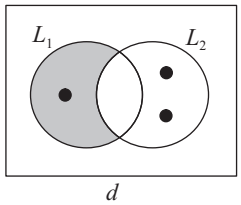

Figure 6: Extending models for a diagram.

Example 5. The diagram in Fig. 6 has a *minimal* model (in the sense that the cardinality of the universal set is minimal) $U = \{1, 2, 3\}$, $\Psi(L_1) = \{1\}$, $\Psi(L_2) = \{2, 3\}$ and, for $i \neq 1, 2$, $\Psi(L_i) = \emptyset$. This model can be used to generate all the models for the diagram. To generate further models, we can add elements to U and we may add these elements to images of contour labels if we so choose. We can also rename the elements in U. As an example, the element 4 can be added to U and we can redefine $\Psi(L_2) = \{2, 3, 4\}$ to give another model for d. No matter what changes we make to the model, we must ensure that the zone $(\{L_1\}, \{L_2\})$ always represents a set containing exactly one element or we will create an interpretation that does not satisfy the diagram.

If a sentence, S, is expressively equivalent to a unitary α-diagram, d, then we will be able to take a minimal model for S and use this model to generate all other models for S in the same manner as above. Given a structure, we will define a *predicate intersection set*. This set is analogous to the image of a zone in an interpretation.

Definition 6. *Let m be a structure and X and Y be finite subsets of $\mathcal{P}$ (the countably infinite set of predicate symbols). Define the* ***predicate intersection set*** *in m with respect to X and Y, denoted $PI(m, X, Y)$, to be*

$$PI(m, X, Y) = \bigcap_{P_i \in X} P_i^m \cap \bigcap_{P_i \in Y} \overline{P_i^m}.$$

We define $\bigcap_{P_i \in \emptyset} P_i^m = \bigcap_{P_i \in \emptyset} \overline{P_i^m} = U$ where U is the domain of m.

In the context of MFOLe, we will identify all of the structures that can be generated from a given structure, m, by adding or renaming elements subject to cardinality restrictions. We will call this class of structures generated by m the *cone* of m. For each sentence, S, we will show that there is a finite set of models, the union of whose cones give rise to only and all the models for S. Central to our approach is the notion of *similar structures with respect to S*. To define similar structures we use the maximum number of *nested quantifiers* in S.

Example 6. Let S be the sentence $\forall x_1\, P_1(x_1) \wedge \forall x_1\, \exists x_2\, \neg(x_1 = x_2)$. The formula $\forall x_1\, P_1(x_1)$ has one nested quantifier and $\forall x_1\, \exists x_2\, x_1 \neq x_2$ has two nested quantifiers. Therefore the maximum number of nested quantifiers in S is two. Now, n nested quantifiers introduce n names, and so it is only possible to talk about (at most) n distinct individuals within the body of the formula. This has the effect of limiting the complexity of what can be said by such a formula. In the particular case here, this observation has the effect that if a model for S has at least two elements in certain predicate intersection sets then S does not place an upper bound on the cardinalities of these predicate intersection sets.

In a model for S, the interpretation of P_1 has to have all the elements, of which there must be either zero or at least two. Also S constrains the predicate intersection set $PI(m, \emptyset, \{P_1\})$ to have cardinality zero. As an example, we consider two models, m_1 and m_2 with domains $U_1 = \{1, 2, 3, 4\}$ and $U_2 = \{1, 2, 5, 6, 7\}$ respectively that are partially defined by $P_1^{m_1} = \{1, 2, 3, 4\}$ and $P_1^{m_2} = \{1, 2, 5, 6, 7\}$. Now

$$|PI(m_1, \emptyset, \{P_1\})| = |\emptyset| = 0 < 2 \quad \text{and} \quad |PI(m_2, \emptyset, \{P_1\})| = |\emptyset| = 0 < 2.$$

Also

$$|PI(m_1, \{P_1\}, \emptyset)| = |U_1| \geq 2 \quad \text{and} \quad |PI(m_2, \{P_1\}, \emptyset)| = |U_2| \geq 2,$$

so S does not place an upper bound on $|PI(m, \{P_1\}, \emptyset)|$. We can think of m_1 and m_2 as extending m_3 with domain $U_3 = \{1, 2\}$ where $P_1^{m_3} = \{1, 2\}$ and $P_j = \emptyset$, for all $j \neq 1$.

Definition 7. *Let S be a sentence and define $q(S)$ to be the maximum number of nested quantifiers in S and define $P(S)$ to be the set of monadic predicate symbols in S. Structures m_1 and m_2 are called* ***similar with respect to*** *S if and only if for each subset X of $P(S)$, either*

1. *$PI(m_1, X, P(S) - X) = PI(m_2, X, P(S) - X)$ or*
2. *$|PI(m_1, X, P(S) - X) \cap PI(m_2, X, P(S) - X)| \geq q(S)$*

and for all subsets Y of $P(S)$ such that $X \neq Y$, $PI(m_1, X, P(S) - X) \cap PI(m_2, Y, P(S) - Y) = \emptyset$. Adapted from [3].

In the previous example, m_1, m_2 and m_3 are all similar with respect to S.

Lemma 3. *Let m_1 and m_2 be similar structures with respect to sentence S. Then m_1 is a model for S if and only if m_2 is a model for S, [3].*

Lemma 3 essentially tells us that any model for a sentence, S, with cardinality greater than $2^{|P(S)|}q(s)$ can be restricted to give another model for S with cardinality at most $2^{|P(S)|}q(s)$.

Definition 8. *Let S be a sentence and m be a model for S. If the cardinality of m is at most $2^{|P(S)|}q(s)$ then we say m is a* ***small model*** *for S. Otherwise we say m is a* ***large model*** *for S.*

Definition 9. *Let S be a sentence and m_1 be a small model for S. The* ***cone*** *of m_1 given S, denoted $cone(m_1, S)$, is a class of structures such that $m_2 \in cone(m_1, S)$ if and only if for each subset X of $P(S)$, there exists an injective map, $f_X \colon PI(m_1, X, P(S) - X) \to PI(m_2, X, P(S) - X)$ which is bijective when $|PI(m_1, X, P(S) - X)| < q(s)$.*

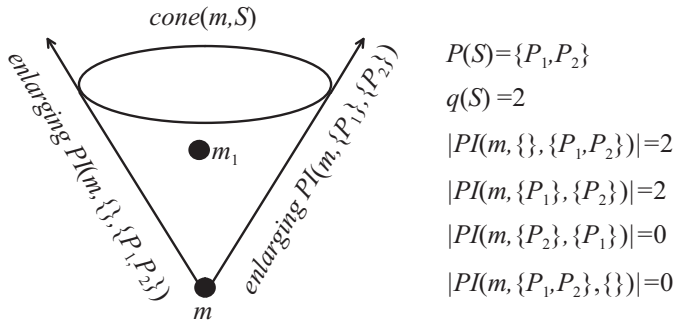

Figure 7: Visualizing cones.

The cone of m given S contains models for S that can be restricted to (models isomorphic to) m. We can think of elements of $cone(m, S)$ as enlarging m in certain 'directions' (adding elements to predicate intersection sets) and 'fixing' m in others (keeping predicate intersection sets the same).

Example 7. Let S be the sentence $\exists x_1 \exists x_2 (P_1(x_1) \vee P_2(x_2))$ and consider $m = \langle \{1,2,3,4\}, =^m, \{1,2\}, \emptyset, \emptyset, ... \rangle$. A visual analogy of $cone(m, S)$ can be seen in Fig. 7. Structure $m_1 = \langle \{1,2,3,4,5,6\}, =^{m_1}, \{1,2,5\}, \emptyset, \emptyset, ... \rangle$ can be obtained from m by enlarging $PI(m, \emptyset, \{P_1, P_2\})$ and $PI(m, \{P_1\}, \{P_2\})$ by adding elements to these sets (and the domain), but keeping $PI(m, \{P_2\}, \{P_1\})$ and $PI(m, \{P_1, P_2\}, \emptyset)$ fixed. Another element of $cone(m, S)$ is the structure $m_2 = \langle \{7,8,9,10\}, =^{m_2}, \{7,8\}, \emptyset, \emptyset, ... \rangle$. Here, m_2 renames the elements in m. The structure $m_3 = \langle \{1,2,3,4\}, =^{m_3}, \{1\}, \emptyset, \emptyset, ... \rangle$ is not in $cone(m, S)$, since there is not an injective map from $PI(m, \{P_1\}, \{P_2\})$ to $PI(m_3, \{P_1\}, \{P_2\})$.

Example 8. Let S be the sentence $\forall x \forall y\, x = y$ and consider the structure $m_1 = \langle \{1\}, =^{m_1}, \emptyset, \emptyset, \emptyset, ... \rangle$ which satisfies S. We have the following cone for m_1:

$$cone(m_1, S) = \{m_2 \in \mathcal{STR} : |PI(m_1, \emptyset, \emptyset)| = |\{1\}| = |PI(m_2, \emptyset, \emptyset)|\}.$$

The class $cone(m_1, S)$ contains only structures that are models for S but does not contain them all, for example $m_3 = \langle \emptyset, \emptyset, ... \rangle$ satisfies S but m_3 is not in $cone(m_1, S)$. All models for S are in the class $cone(m_1, S) \cup cone(m_3, S)$. In this sense, m_1 and m_3 classify all the models for S. We can draw a diagram expressively equivalent to S using information given by m_1 and m_3. This diagram is a disjunction of two unitary diagrams, shown in Fig 8. The unitary diagram arising from m_1 has one spider, no contours and is entirely shaded. That arising from m_3 has no spiders, no contours and is entirely shaded.

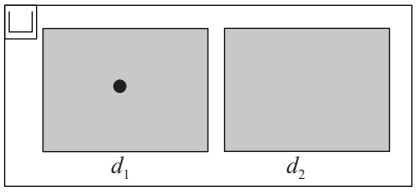

Figure 8: A diagram expressively equivalent to $\forall x \forall y\, x = y$.

We will show that, given a sentence, S, there is a finite set of small models, the union of whose cones give rise to only and all the models for S. We are able to use these models to identify a diagram expressively equivalent to S. In order to identify such a finite set we require the notion of *partial isomorphism* between structures.

Definition 10. *Let m_1 and m_2 be structures with domains U_1 and U_2 respectively. Let Q be a set of monadic predicate symbols. If there exists a bijection $\gamma : U_1 \to U_2$ such that*

$$\forall P_i \in Q\, \forall x \in U_1\ x \in P_i^{m_1} \Leftrightarrow \gamma(x) \in P_i^{m_2}$$

*then m_1 and m_2 are **isomorphic restricted to Q** and γ is a **partial isomorphism**.*

If m_1 and m_2 are isomorphic restricted to $P(S)$ then m_1 is a model for S if and only if m_2 is a model for S. Also, there are finitely many small models for sentence S, up to isomorphism restricted to $P(S)$.

Definition 11. *Let S be a sentence. A set of small models, $c(S)$, for S is called a* ***classifying set of models*** *for S if for each small model, m_1, for S there is a unique m_2 in $c(S)$ such that m_1 and m_2 are isomorphic, restricted to $P(S)$.*

Theorem 3. *Let $c(S)$ be a classifying set of models for sentence S. Then $c(S)$ is finite and $\bigcup_{m \in c(S)} cone(m, S)$ contains all and only models for S.*

Definition 12. *Let m be a small model for sentence S. The unitary α-diagram, d,* ***representing*** *m, is defined as follows.*

1. *The contour labels arise from the predicate symbols in $P(S)$: $L(d) = \{L_i \in \mathcal{L} : \exists P_i \in \mathcal{P}\ P_i \in P(S)\}$.*
2. *The diagram has no missing zones: $Z(d) = \{(a, b) : a \subseteq L(d) \wedge b = L(d) - a\}$.*
3. *The shaded zones in d are given as follows. Let X be a subset of $P(S)$ such that $|PI(m, X, P(S) - X)| < q(S)$. The zone (a, b) in $Z(d)$ where $a = \{L_i : P_i \in X\}$ is shaded.*
4. *The number of spiders in each zone is the cardinality of the set $|PI(m_1, X,\ P(S) - X)|$ where X gives rise to the containing set of contour labels for that zone. The set of spider identifiers is then given by:*

$$SI(d) = \{(n, r) : \exists X\ X \subseteq P(S) \wedge |PI(m, X, P(S) - X)| > 0 \wedge \\ n = |PI(m, X, P(S) - X)| \wedge r = \{(a, b) \in Z(d) : a = \{L_i : P_i \in X\}\}\}.$$

We write $\mathcal{REP}(m_1, S) = d$. Let $c(S)$ be a set of classifying models for S. Define $\mathcal{D}(S)$ to be a disjunction of unitary diagrams, given by $\mathcal{D}(S) = \bigsqcup_{m \in c(S)} \mathcal{REP}(m, S)$, unless $c(S) = \emptyset$, in which case $\mathcal{D}(S) = \perp$.

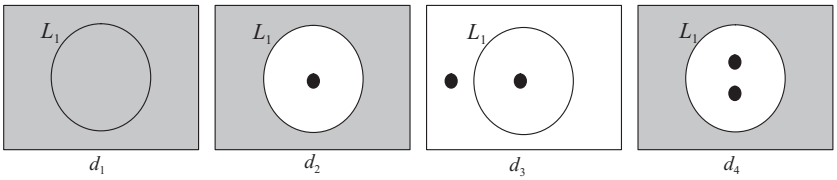

Figure 9: Constructing diagrams from models.

Example 9. Let S be the sentence $\exists x_1 P_1(x_1) \vee \forall x_1 P_1(x_1)$. To find a classifying set of models we must consider structures of all cardinalities up to $2^{|\{P_1\}|} \times q(S) = 2^1 \times 1 = 2$. There are six distinct structures (up to isomorphism restricted to $P(S)$) with cardinality at most 2. Precisely four of these structures are models for S and are listed below.

1. $m_1 = \langle \emptyset, \emptyset, ... \rangle$,
2. $m_2 = \langle \{1\}, =^{m_2}, \{1\}, \emptyset, \emptyset, ... \rangle$,
3. $m_3 = \langle \{1, 2\}, =^{m_3}, \{1\}, \emptyset, \emptyset, ... \rangle$,
4. $m_4 = \langle \{1, 2\}, =^{m_4}, \{1, 2\}, \emptyset, \emptyset, ... \rangle$.

Therefore, the class $cone(m_1, S) \cup cone(m_2, S) \cup cone(m_3, S) \cup cone(m_4, S)$ contains only and all the models for S. We use each of these models to construct a diagram. Models m_1, m_2, m_3 and m_4 give rise to d_1, d_2, d_3 and d_4 in Fig. 9 respectively. The diagram $d_1 \sqcup d_2 \sqcup d_3 \sqcup d_4$ is expressively equivalent to S. This is not the 'natural' diagram one would associate with S.

Theorem 4. *Let S be a sentence and $c(S)$ be a set of classifying models for S. Then S is expressively equivalent to $\mathcal{D}(S)$.*

Hence the language of spider diagrams and MFOLe are equally expressive.

4 Constraint Diagrams

The constraint diagram system we introduce here is a fragment of the more expressive full constraint diagram notation (which includes further syntactic elements) introduced by Kent [18]. This fragment is considerably more expressive than the spider diagram system and the work represents a significant step towards a reasoning system based on the full notation. We give the syntax and semantics of our restricted constraint diagrams and state syntactic criteria that are sufficient for identifying the satisfiability of some constraint diagrams. Some of the reasoning rules defined in the thesis will be illustrated and we sketch the completeness proof strategy.

4.1 Syntax

The formal (abstract) syntax for constraint diagrams modifies and extends that given for spider diagrams. Here, we will informally sketch the syntax rather than including all of the formal definitions which are given in the thesis.

A *unitary constraint diagram* is a unitary spider diagram with, possibly, additional syntactic components as outlined below. Constraint diagrams may contain another type of spider, called *universal spiders* which are represented by asterisks. In this system, universal spiders inhabit single zones only, with at most one in any given zone. In the full constraint diagram language, universal spiders may inhabit many zones and any finite number of universal spiders may have any given habitat. For clarity, non-universal spiders will now be called *existential spiders*. We may also place *derived contours* in unitary diagrams. A derived contour is a contour without a label. In this system, the inside of a derived contour is always shaded and they are restricted to representing the empty set. In the full constraint diagram language, derived contours represent other sets as well. The 'special' derived contour in this restricted system simplifies some of the details in the completeness proof. *Arrows* can be placed in unitary diagrams. Each arrow has a label, a source and a target. Just as for contour labels, it is of use to define $\mathcal{AL}$ to be a countably infinite set of **arrow labels**. An **arrow** is a triple (l, s, t) where l is an arrow label, s is the source which can be a spider and t is the target which can be either an existential spider of a contour (possibly derived). In a unitary diagram, there is at most one derived contour, which must be the target of at least one arrow. In the full constraint diagram language, unitary diagrams can contain finitely many derived contours, all of which are the target of some arrow. Furthermore, in our system, each universal spider in a unitary diagram must be the source of at least one arrow.

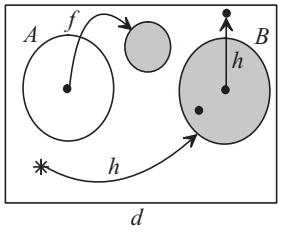

Figure 10: A constraint diagram with a derived contour.

Example 10. The diagram in figure 10 contains three contours. One of these contours is derived since it has no label. There are four existential spiders. One of these is the target of an arrow and two are sources of arrows. Two of the three arrows in the diagram have the same label, h. The arrow with label f targets the derived contour. By convention, the derived contour does not affect the set of zones: the zones of d are the same as those for the underlying spider diagram.

We define, for unitary constraint diagram d, the set of existential spiders in d to be $ES(d)$ and we denote the habitat of an existential spider, e, in d by $\eta(e)$. As with spider diagrams, $\perp$ is a unitary diagram and we can join constraint diagrams using the logical connectives $\sqcup$ and $\sqcap$.

4.1.1 Semantics

Constraint diagrams that do not contain any arrows (and, therefore, no universal spiders or derived contours) are spider diagrams. The semantics of constraint diagrams extend the semantics of their underlying spider diagrams. Arrow labels represent binary relations. An arrow, together with its source and target, represents a property of the relation represented by its label. A universal spider represents universal quantification over the set represented by its habitat.

The diagram in Fig. 11 expresses that (the sets represented by) A and B are disjoint, B is not empty and there is an x in $U-(A\cup B)$ such that for all a in A the relational image of a under f is x. This diagram could also be interpreted as 'for all a in A, there exists an x in $U-(A\cup B)$ such that the relational image of a under f is x', but we will not allow such a reading. To avoid ambiguity in diagram reading and to make the system tractable, we restrict the semantic interpretation so that 'there exists' takes precedence over 'for all'. Relaxing this semantic constraint has non-trivial outcomes.

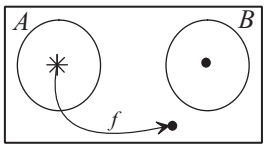

Figure 11: A constraint diagram.

Definition 13. *An **interpretation** of $\mathcal{CL}$ and $\mathcal{AL}$ is a triple $m=(U,\Psi,\phi)$ where (U,Ψ) is an interpretation for the spider diagram system (that is, as in definition 2) and $\phi\colon \mathcal{AL}\to\mathbb{P}(U\times U)$ is a function mapping arrow labels to relations on U.*

Next we formalize the notion of the image of a relation. Let R be a relation on a set U. Define the **image** of R to be $im(\mathsf{R})=\{b\in U:(a,b)\in\mathsf{R}\}$. Let A be a subset of U. Define $A.\mathsf{R}$ to be $A.\mathsf{R}=im(\mathsf{R}\cap(A\times U))$ and say $A.\mathsf{R}$ is the image of R with the domain restricted to A.

As for spider diagrams, for each unitary diagram d and for each region $r\subseteq Z(d)$ we define $S(r,d)$ and $T(r,d)$ to be the set of existential spiders that are completely within r and that touch r respectively (that is, $S(r,d)$ and $T(r,d)$ do not contain universal spiders). We define $A_e(d)$ and $A_u(d)$ to be the sets of arrows in d with an existential source and universal source respectively; the sets $A_e(d)$ and $A_u(d)$ partition the set of arrows in d, denoted $A(d)$. Arrows in $A_e(d)$ and $A_u(d)$ are called existential arrows and universal arrows respectively.

Definition 14. *Let d be a constraint diagram and let (U,Ψ,ϕ) be an interpretation. The **semantics predicate**, $P_d(m)$, of d is defined as follows. If d is unitary and $d\neq\perp$ then $P_d(m)$ is the conjunction of the following conditions.*

(i) ***Plane Tiling Condition.*** *All elements are in sets represented by zones: $\bigcup_{z\in Z(d)}\Psi(z)=U$.*

(ii) There exists an extension of $\Psi\colon\mathcal{L}\cup\mathcal{Z}\cup\mathcal{R}\to\mathbb{P}U$ to $\Psi\colon\mathcal{L}\cup\mathcal{Z}\cup\mathcal{R}\cup ES(d)\to\mathbb{P}U$ such that the conjunction of the following conditions are satisfied.

(a) ***Spiders Condition.*** *Each existential spider represents the existence of an element in the set represented by its habitat:*

$$\forall\, e\in ES(d)\,|\Psi(e)|=1\wedge\Psi(e)\subseteq\Psi(\eta(e)).$$

(b) ***Strangers Condition.*** *No two existential spiders represent the existence of the same element:*

$$\forall\, e_1,e_2\in ES(d)\,\Psi(e_1)=\Psi(e_2)\Rightarrow e_1=e_2.$$

(c) ***Shading Condition.*** *Each shaded zone, z, represents a subset of the elements represented by the existential spiders touching z:*

$$\forall\, z\in Z^*(d)\,\Psi(z)\subseteq\bigcup_{e\in T(\{z\},d)}\Psi(e).$$

(d) ***Existential Arrows Condition.*** *For any existential arrow, (l,s,t), the image of $\phi(l)$ with its domain restricted to $\Psi(s)$ equals $\Psi(t)$:*

$$\forall\,(l,s,t)\in A_e(d)\,\Psi(s).\phi(l)=\Psi(t).$$

(e) ***Universal Arrows Condition.*** *For any universal arrow, (l,s,t), the image of $\phi(l)$ with its domain restricted to any element in the set represented by the habitat of s equals $\Psi(t)$:*

$$\forall\,(l,s,t)\in A_u(d)\,\forall\, x\in\Psi(\eta(s))\,\{x\}.\phi(l)=\Psi(t).$$

Otherwise, $P_d(m)$ is defined in a similar manner to the spider diagram case.

4.2 Satisfiability

Unlike spider diagrams, not all unitary constraint diagrams are satisfiable. For example, the diagram in Fig. 12 is unsatisfiable. It expresses that there is an element in A that is related to exactly one element, x say, in $U - A$ under the relation f and exactly one element in $U - A$ distinct from x also under the relation f, which cannot happen. We give syntactic criteria for identifying whether a unitary α-diagram is satisfiable.

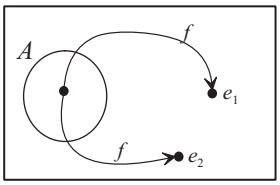

Figure 12: An unsatisfiable unitary constraint diagram.

Definition 15. *For each arrow, $a = (l, s, t)$ in unitary α-diagram d, define the set of* ***hit existential spiders*** *of a in d, denoted $hit(a, d)$, to be*

(i) $hit(a, d) = \emptyset$ *if t is derived,*

(ii) $hit(a, d) = \{t\}$ *if $t \in ES(d)$ and*

(iii) $hit(a, d) = S(t, d)$ *if $t \in L(d)$,*

where, for contour (label) t, $S(t, d)$ is the set of existential spiders whose habitats are completely within t. The arrows of d are ***pairwise compatible*** *if and only if*

(i) every pair of arrows with the same existential spider as their source and the same label have the same hits and

(ii) every pair of arrows with the same universal spider, s, as their source and the same label have the same hits or the habitat of s is not inhabited by any existential spider and

(iii) if an existential spider has the same habitat as a universal spider and both are the source of arrows with the same label then these arrows have the same hits.

If the arrows of d are not pairwise compatible then d is said to contain ***incompatible arrows****.*

In Fig. 12 one arrow with label f has hit $\{e_1\}$ and the other has hit $\{e_2\}$. Since these arrows have the same source they are incompatible, failing condition (ii). Incompatible arrows provide the only source of unsatisfiability for unitary α-diagrams.

Theorem 5. *Unitary α-diagram d ($\neq \perp$) is satisfiable if and only if the arrows of d are pairwise compatible.*

Part of the strategy to prove the above theorem is to construct a model for d with pairwise compatible arrows as follows. The universal set is taken to be $ES(d)$, the set of existential spiders in d. Each contour label, $l \in \mathcal{L}$, is mapped to

$$\Psi(l) = \{e \in ES(d) : e \text{ is placed inside } l\}.$$

To interpret the arrow labels we firstly define $US(d)$ to be the set of universal spiders in d and, as for existential spiders, we denote the habitat of universal spider u by $\eta(u)$. Each arrow label, $l \in \mathcal{AL}$, then maps to

$$\phi(l) = \{(e_1, e_2) \in ES(d) \times ES(d) : \exists (l, s, t) \in A(d)\,(e_1 = s \vee (s \in US(d) \wedge \eta(e_1) = \eta(s))) \wedge e_2 \in hit((l, s, t), d)\}.$$

This interpretation plays an important role in the completeness proof.

4.3 Reasoning Rules for Unitary Diagrams

In this section we illustrate some of the reasoning rules for constraint diagrams. Due to space reasons, the majority of the rules are not included. Those discussed here are stated informally and illustrate the types of rules that are in the system. In the thesis, the rules range from being very simple to define (such as the erasure of an arrow rule below) to having more involved definitions (such as the disjunctifying unitary α-diagrams rule below). Even though their formal

definitions are not given, the rules included here indicate the breadth of difficulty levels associated with defining the rules. In the thesis, all of the rules are formally defined using the abstract syntax. In total, there are twenty five rules, of which twenty two are required for completeness. The other three rules provide useful shortcuts for certain sequences of rule applications. Some of the reasoning rules for constraint diagrams are modifications and extensions of those for spider diagrams and, in addition, many new rules relating to arrows are necessary for completeness. For example, in Fig. 13, the arrow labelled f can erased from the diagram d_1 to give d_2.

Rule 1. ***Erasure of an arrow.*** *We may erase any arrow from a unitary diagram. If erasing an arrow results in a universal spider that is no longer the source of an arrow then that spider is also erased. Similarly, if erasing an arrow results in a derived contour that is no longer the target of an arrow then that contour is also erased.*

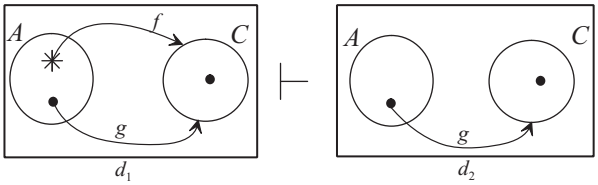

Figure 13: An application of rule 1.

Erasing an arrow (potentially) loses information, as do some other rules in the system. The remaining rules discussed here preserve semantic information.

Rule 2. ***Introduction of a contour.*** *A contour may be drawn in the interior of the boundary rectangle of a unitary diagram provided the following occurs.*

1. *The new contour has a label not present in the diagram.*
2. *Each zone splits into two zones and shading is preserved.*
3. *Each node of an existential spider (recall, an existential spider is a graph) is replaced by a connected pair of nodes – one in each new zone of the habitat.*
4. *Each universal spider, u, is replaced by a pair of universal spiders – one in each new zone of the habitat. Each arrow, a, sourced on u is replaced by a pair of arrows with the same label and target as a, one sourced on each new universal spider.*

Fig.14 shows an application of rule 2.

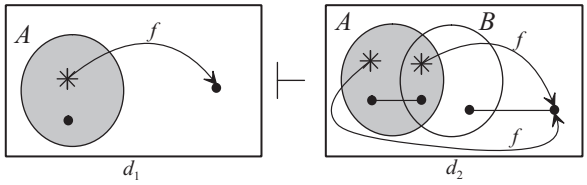

Figure 14: An application of rule 2.

Rule 3. ***Introduction of an arrow: universal equivalence.*** *Let d be a unitary diagram with a shaded zone z where every existential spider that touches z is the source of an arrow with label l and target t (l and t are fixed). Then we can introduce a universal arrow (and if necessary a universal spider) whose source inhabits z, labelled l with target t provided that the new arrow is not already present in d.*

In Fig. 15, the diagram d_1 expresses that each element in A has relational image, under f, that is B. Therefore a universal spider can be introduced to the zone inside A, which is the source of an arrow with label f, targeted on B.

Rule 4. ***Introduction of an arrow: contour to contour.*** *Let d be a unitary diagram with a pair of contours, C_1 and C_2, whose symmetric difference is shaded and not touched by any existential spider and C_1 is the target of an arrow, a. Then we can introduce an arrow to d with the same source and label as a and target C_2 provided that the new arrow is not already present in d.*

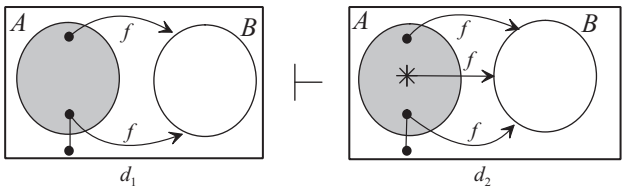

Figure 15: An application of rule 3.

The diagram d_1 in Fig. 16 expresses that there is an element, x say, in A that is related to each element in B under f. Furthermore d expresses that B represents the same set as C. Thus x is related to each element in C under f.

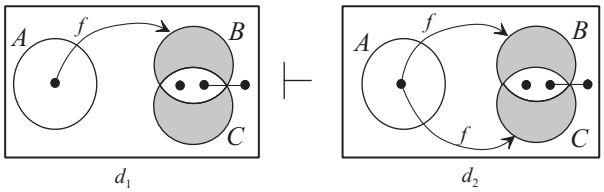

Figure 16: An application of rule 4.

In total there are six rules that introduce an arrow to a unitary diagram.

Rule 5. ***Excluded middle for regions.*** *Let d be a unitary diagram with a completely non-shaded region, r. We can replace d with $d_1 \sqcup d_2$, where d_1 and d_2 are copies of d except that r is shaded in d_1 and r contains an additional existential spider in d_2.*

The excluded middle for regions rule is applied to diagram d in Fig. 17. We shade $B - C$ (giving d_1) and add an existential spider to $B - C$ (giving d_2), as shown in $d_1 \sqcup d_2$.

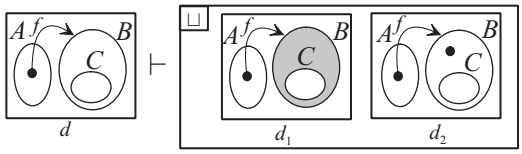

Figure 17: An application of rule 5.

Rule 6. ***Excluded middle for arrows.*** *Let d be a unitary α-diagram such that every zone in d is shaded and, for each subset of $ES(d)$, E_i, such that $|E_i| > 1$, there is a contour, A, such that $S(A, d) = E_i$. Let $l \in \mathcal{AL}$ and let e be an existential spider in d that is not the source of an arrow with label l. Define $\mathcal{E}(d, l)$ to be the set of unitary diagrams, d_j, each of which is a copy of d except that d_j contains an additional arrow with source e, label l and any target. Then we may replace d with $\bigsqcup_{d_j \in \mathcal{E}(d,l)} d_j$.*

An application of this rule is illustrated in Fig. 18. Since every possible subset of U is explicitly represented, we can deduce that any given element must be related to nothing, itself, the other element or both elements under the relation l, shown explicitly by $d_1 \vee d_2 \vee d_3 \vee d_4$.

The final rule that we illustrate allows us to replace a diagram with a disjunction of unitary diagrams. There is a spider diagram version of this rule (called 'combining') that is essential to the completeness proof strategy used in the spider diagram system. The combining rule replaces two unitary α-diagrams taken in conjunction by a single unitary α-diagram. To extend the proof strategy to this constraint diagram system, we require a constraint diagram version of this rule and we call the rule *disjunctifying unitary α-diagrams*. The basic operation of disjunctification is performed on unitary α-diagrams which have the same sets of zones. Unlike the combining rule in the spider diagram system, disjunctifying constraint diagrams may result in a disjunction of unitary α-diagrams. However, we first we consider a

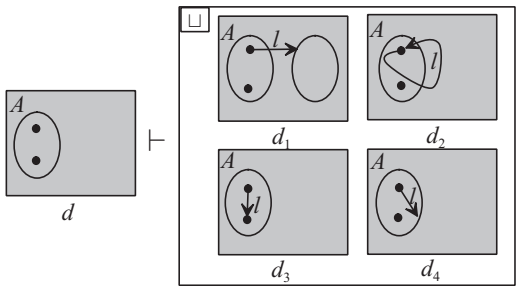

Figure 18: An application of rule 6.

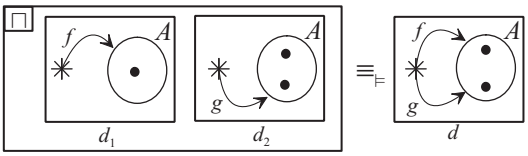

Figure 19: Disjunctifying constraint diagrams.

simple example where disjunctifying two unitary diagrams does give a unitary diagram: the diagram $d_1 \sqcap d_2$ in Fig. 19 is semantically equivalent to the unitary diagram d.

For our second example, the diagrams $d \sqcap d'$ and $d_1 \sqcup d_2 \sqcup d_3 \sqcup d_4$ in Fig. 20 are semantically equivalent. In $d \sqcap d'$ the spiders inside A could represent the same element or distinct elements. Similarly for B. This pair of choices gives four alternatives, each represented by one of d_1, d_2, d_3 and d_4.

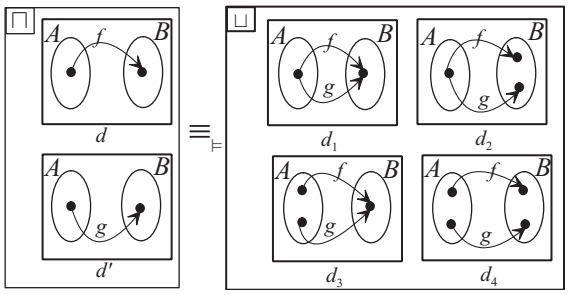

Figure 20: Disjunctifying constraint diagrams.

To define the disjunctification of two unitary α-diagrams we first identify the unitary components that form the disjunctification, called partial combinations.

Definition 16. *Let d_0 and d_1 be two unitary α-diagrams with the same zone sets. Let d be a unitary α-diagram that does not contain incompatible arrows. Then d is called a* ***partial combination*** *of $d_0 \sqcap d_1$ if and only if each of the following are satisfied.*

(i) The diagram d has the same zone set as d_0 and d_1.

(ii) The shaded zones in d are precisely those which are shaded in at least one of d_0 and d_1.

(iii) The number of existential spiders in any shaded zone in d is the maximum number of existential spiders inhabiting that zone in d_0 and d_1.

(iv) The number of existential spiders inhabiting any unshaded zone in d is at most the maximum of

(a) the number of existential spiders inhabiting that zone in d_0,

(b) the number of existential spiders inhabiting that zone in d_1,

(c) the sum total of the number of existential spiders that are sources or targets of arrows inhabiting that zone in d_0 and d_1.

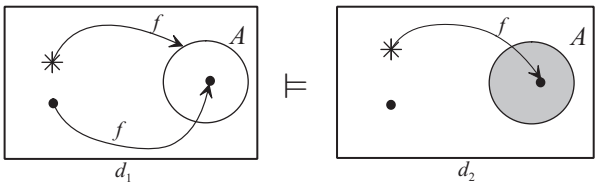

Figure 21: The diagram d_2 is a logical consequence of d_1.

(v) The number of existential spiders in any zone in d is at least the largest number of existential spiders inhabiting that zone in d_0 and d_1.

(vi) There is a universal spider in a zone in d whenever there is a universal spider in that zone in d_0 or d_1.

(vii) All of the arrows in d_0 occur in d, similarly for d_1, and no others.

We define $\mathcal{D}_{pc}(d_0 \sqcap d_1)$ to be the set of partial combinations of $d_0 \sqcap d_1$.

It may be that $\mathcal{D}_{pc}(d_0 \sqcap d_1) = \emptyset$, for example, if one of d_0 and d_1 contains incompatible arrows.

Definition 17. *Let d_0 and d_1 be unitary α-diagrams such that $Z(d_0) \equiv Z(d_1)$ or $d_0 = \bot$ or $d_1 = \bot$. Define the **disjunctification** of d_0 and d_1, denoted $d_0 * d_1$, as follows.*

1. *If $d_0 = \bot$ or $d_1 = \bot$ then $d_0 * d_1 = \bot$.*
2. *If a zone in one diagram contains more existential spiders than in a corresponding shaded zone in the other diagram then $d_0 * d_1 = \bot$.*
3. *If $\mathcal{D}_{pc}(d_0 \sqcap d_1) = \emptyset$ then $d_0 * d_1 = \bot$.*
4. *Otherwise $d_0 * d_1 = \bigsqcup_{d \in \mathcal{D}_{pc}(d_0 \sqcap d_1)} d$.*

To summarize, $d_0 * d_1 \equiv_{\vDash} d_0 \sqcap d_1$ and $d_0 * d_1$ is a disjunction of unitary α-diagrams.

Rule 7. ***Disjunctifying unitary α-diagrams.*** *Let d_0 and d_1 be unitary α-diagrams such that $Z(d_0) = Z(d_1)$ or $d_0 = \bot$ or $d_1 = \bot$. Then $d_0 \sqcap d_1$ may be replaced by $d_0 * d_1$.*

Of the twenty five reasoning rules defined in the thesis, seventeen are 'diagrammatic rules' and the remaining eight are 'propositional logic' rules.

4.4 Soundness and Completeness

In the thesis, it is shown that all of the rules are valid (most of the validity proofs are straightforward, but some are more involved) and that the system is sound.

Theorem 6. ***Soundness.*** *Let d_1 and d_2 be constraint diagrams. If $d_1 \vdash d_2$ then $d_1 \vDash d_2$.*

The strategy for proving completeness of the spider diagram system extends to this constraint diagram system. Part of the completeness proof strategy used for spider diagrams begins with a disjunction of unitary α-diagrams each with the same zone sets, (acquired by adding contours and zones – using an 'add shaded zone' rule – then splitting the spiders and, finally, disjunctifying). For spider diagrams, there are simple syntactic checks that establish whether one unitary α-diagram is a logical consequence of another. If we consider the subset of all spider diagrams consisting of unitary α-diagrams with the same zone sets, then all the rules necessary for completeness are erasure rules.

In this constraint diagram system it is possible to transform any diagram into a disjunction of unitary α-diagrams, using the same strategy. However, for constraint diagrams, it is not easy to determine whether one unitary α-diagram is a logical consequence of another. There are examples of unitary α-diagrams with the same label sets, where one constraint diagram, d_2 say, is a logical consequence of another but requires more complex rules than simple erasure of components to establish syntactic entailment. An example of two such diagrams is given in Fig. 21.

Central to the completeness proof for constraint diagrams is the notion of a *β-diagram*. A **β-diagram** is an α-diagram in which every zone is either shaded or inhabited by an existential spider. We can apply the the excluded middle for regions rule to a disjunction of unitary α-diagrams, adding shading and spiders to produce a disjunction of

unitary β-diagrams. If we introduce 'all possible' syntactic elements to a unitary β-diagram, d_1, using our reasoning rules, giving d_2, then any unitary β-diagram, d_3, that is a logical consequence of d_1 will 'contain only syntactic elements that are in d_2'. It can be shown that d_3 is obtainable from d_1 by simply erasing components of d_2. This gives a completeness result for unitary β-diagrams[3]. We note that the proof of this completeness result for unitary β-diagrams is particularly challenging and requires non-trivial and subtle insights.

We use the completeness result for unitary β-diagrams to prove completeness of the system. Consider two constraint diagrams that satisfy $d_1 \models d_2$. All of the rules that transform a constraint diagram into a disjunction of unitary β-diagrams are equivalences, so we transform d_1 and d_2 into such disjunctions, ${}^{\beta}d_1$ and ${}^{\beta}d_2$ respectively, as outlined above. We then apply the excluded middle for regions rule repeatedly to ${}^{\beta}d_1$ until the number of existential spiders in each zone of each unitary component exceeds the number of existential spiders in that zone in any unitary component of ${}^{\beta}d_2$, giving a diagram d, say. We then add 'all possible' syntactic elements to d using our reasoning rules giving diagram d', say. We can then show each unitary component of d', say d'', semantically entails a unitary component of ${}^{\beta}d_2$, say d_i. From the completeness result for unitary β-diagrams, $d'' \vdash d_i$ and it follows that $d_1 \vdash d_2$.

The sketch of the proof strategy that we have given does not convey the difficulty of the proof. The proof, along with a more detailed discussion and illustrative examples, can be found in the thesis on pages 196 – 277.

Theorem 7. ***Completeness.*** *Let d_1 and d_2 be constraint diagrams. If $d_1 \models d_2$ then $d_1 \vdash d_2$.*

To prove completeness, we constructed an algorithm to transform d_1 into d_2. It is straightforward to modify this algorithm to detect whether $d_1 \models d_2$, giving us not only a decidability result but also a decision procedure.

Theorem 8. ***Decidability.*** *Let d_1 and d_2 be constraint diagrams. There is an algorithm which determines whether $d_1 \vdash d_2$.*

5 Conclusions

There are three main contributions made in this thesis. First, we developed a spider diagram system that directly extends previous work, particularly that in [20]. The spider diagram system presented in this thesis is the 'definitive' version, in that it is based on Euler diagrams and does not restrict to CNF. The approaches we have taken in our formalization are more elegant than those for previous systems. If we had, instead, chosen to closely follow previous approaches then many of the formal definitions and proofs would be considerably more complex.

The two most significant contributions are establishing the expressive power of spider diagrams and developing the first ever constraint diagram reasoning system. Shin's approach to proving that Venn-II is equivalent in expressive power to MFOL [22] does not extend to the spider diagram case. As with Shin's approach, similar work on converting MFOL sentences into diagrams relies on syntactically manipulating MFOL sentences until they can easily be translated into a diagram (where possible), see for example [23]. Thus, a new strategy needed to be found and we took a very different approach, using a model theoretic argument. By proving that each MFOLe sentence, S, has a finite set of classifying models, we were able to construct a spider diagram expressing the same information as S. As a consequence of this equivalence in expressive power, along with the sound and complete set of reasoning rules, we can choose to use spider diagrams to prove theorems of theories whose axioms are MFOLe sentences.

The development of the sound and complete constraint diagram system in this thesis represents a significant step towards developing a reasoning system based on the full constraint diagram langauge which has potentially important applications. We have shown that it is possible to develop highly expressive, sound and complete diagrammatic logics. Since this thesis was completed, a formalization of the syntax and semantics of the full constraint diagram notation has been provided [4].

A very interesting question is "what fragments of the full constraint diagram language yield decidable systems?" In this thesis we have presented such a fragment. Ideally, a large fragment of the constraint diagram language that yields a decidable system can be identified: users of the language, for example software engineers, need to know whether their constraints are satisfiable. A known result in first order predicate logic is that the satisfiability problem for sentences in the Bernays-Schönfinkel-Ramsey class with equality is decidable [1]. These sentences are prenex formulas of the form $\exists x_1 \exists x_2 ... \exists x_n \forall x_{n+1} \forall x_{n+2} ... \forall x_{n+m} F$, that is 'there exists' takes precedence over 'for all'. Our constraint diagram system may well be equivalent in expressive power to a fragment of this class, since we read existential spiders before universal spiders. However, a direct translation of our diagrams into FOPL (the translation is rather more complex than for spider diagrams) does not yield sentences in prenex normal form. Some of the equivalences that are required to transform FOPL sentences into prenex normal form do not hold in the empty structure, which we allow. Thus we

[3]The outline of the strategy used to prove completeness for unitary β-diagrams is not entirely accurate, but illustrates the essentials of the argument.

cannot immediately deduce that our constraint diagram system is equivalent in expressive power to a fragment of the Bernays-Schönfinkel-Ramsey class with equality.

The reasoning systems we have defined in this thesis provide the necessary mathematical underpinning required to implement automated reasoning tools. Already one theorem proving tool, available from [25], has been implemented based on the spider diagram language we have developed [7]. The algorithm that this tool uses to find a proof that one diagram semantically entails another or to provide a counterexample is an improved version of that which can be derived from the completeness proof strategy used for the spider diagram system. The improvements reduce the number of rule applications made by using a sophisticated approach when comparing disjunctions of unitary diagrams. However, many of the proofs generated by this algorithm are unnecessarily long.

In [5, 6] a heuristic approach to generating proofs in our spider diagram language is presented. Measures between diagrams are defined that combine to give a lower bound on the number of proof steps required to transform any given premise into any given conclusion. Taking this lower bound as a heuristic function and the number of reasoning rules applied as a cost function, the A^* search algorithm is applied to find a shortest proof between two diagrams. The method guarantees to find a shortest proof from the premise to the conclusion, provided a proof exists and there is sufficient time and computer memory.

It should be possible to extend the theorem prover to the constraint diagram language that we have defined. Certainly, extending the tool to incorporate the additional syntax and reasoning rules is feasible. For automated theorem proving, it is possible to extract a direct proof writing algorithm from the completeness proof for our constraint diagram system. With improvements to its efficiency, this algorithm could be of practical use.

In summary, we have developed diagrammatic logics that can be used to formally specify software systems, see [16, 19] for some modelling examples. The completeness proof for the constraint diagram system was particularly challenging and many subtle arguments contribute to the proof. The strategy we have taken to prove the expressiveness result for spider diagrams is novel, requiring deep insights into the model-theoretic relationships between MFOLe sentences and spider diagrams. The hope is that diagrammatic logics will be more appealing to software engineers (and others) who do not like using textual logics for formal specification. Indeed, diagrams are likely to facilitate communication between all of the stakeholders, including domain experts and managers who are unlikely to be familiar with textual logics. We believe that the research presented in this thesis, along with related work, will help to promote the use of logic in computing applications. The work in the thesis represents a significant advance in this area.

References

[1] E. Borger, E' Gradel, and E. Gurevich. *The Classical Decision Problem.* Springer-Verlag, 1997.

[2] L. Choudhury and M. K. Chakraborty. On extending Venn diagrams by augmenting names of individuals. In *Proceedings of Diagrams 2004, Cambridge, UK*, LNAI, pages 142–146. Springer-Verlag, March 2004.

[3] B. Dreben and D. Goldforb. *The Decision Problem. Solvable Classes of Quantificational Formulas.* Addison Wesley Publishing Company Inc., 1979.

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

[5] J. Flower, J. Masthoff, and G. Stapleton. Generating proofs with spider diagrams using heuristics. In *Visual Languages and Computing*, pages 279–285. Knowledge Systems Insitute, September 2004.

[6] 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, Diagrams 2004, Cambridge, UK*, pages 166–181. Springer-Verlag, 2004.

[7] J. Flower and G. Stapleton. Automated theorem proving with spider diagrams. In *Proceedings of Computing: The Australsain Theory Symposium (CATS'04), Dunedin, New Zealand*, volume 91 of *ENTCS*, pages 16–132. Science Direct, January 2004.

[8] 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.

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

[10] E. Hammer and S. J. Shin. Euler's visual logic. *History and Philosophy of Logic*, pages 1–29, 1998.

[11] J. Howse, F. Molina, S-J. Shin, and J. Taylor. On diagram tokens and types. In *Proceedings of Diagrams 2002, Callaway Gardens, Georgia, USA*, pages 146–160. Springer-Verlag, April 2002.

[12] J. Howse, F. Molina, and J. Taylor. On the completeness and expressiveness of spider diagram systems. In *Proceedings of Diagrams 2000, Edinburgh, UK*, pages 26–41. Springer-Verlag, September 2000.

[13] J. Howse, F. Molina, and J. Taylor. SD2: A sound and complete diagrammatic reasoning system. In *Proceedings VL 2000: IEEE Symposium on Visual Languages, Seattle, USA*, pages 127–136. IEEE Computer Society Press, 2000.

[14] J. Howse, F. Molina, J. Taylor, and S. Kent. Reasoning with spider diagrams. In *Proceedings of IEEE Symposium on Visual Languages (VL99), Tokyo*, pages 138–147. IEEE Computer Society Press, September.

[15] J. Howse, F. Molina, J. Taylor, S. Kent, and J. Gil. Spider diagrams: A diagrammatic reasoning system. *Journal of Visual Languages and Computing*, 12(3):299–324, June 2001.

[16] J. Howse and S. Schuman. Precise visual modelling. *Software and Systems Modelling*, 4:310–325, 2005.

[17] C. John. Reasoning with projected contours. In *Proceedings of 3rd International Confernece, Diagrams 2004, Cambridge, UK*, pages 147–150. Springer-Verlag, 2004.

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

[19] 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.

[20] F. Molina. *Reasoning with extended Venn-Peirce diagrammatic systems.* PhD thesis, University of Brighton, 2001.

[21] O. Patrascoiu, S. Thomspon, and P. Rodgers. Tableaux for diagrammatic reasoning. In *Visual Languages and Computing*, pages 279–286. Knowledge Systems Insitute, September 2005.

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

[23] N. Swoboda and G. Allwein. Using DAG transformations to verify Euler/Venn homogeneous and Euler/Venn FOL heterogeneous rules of inference. *Journal of Software and System Modeling*, 3(2):136–149, May 2004.

[24] N. Swoboda and J. Barwise. The information content of Euler/Venn diagrams. In *Proceedings LICS workshop on Logic and Diagrammatic Information*, 1998.

[25] Visual Modelling Group: `http://www.it.bton.ac.uk/research/vmg`.