What Can Spider Diagrams Say?

Gem Stapleton[1], John Howse[1], John Taylor[1], and Simon Thompson[2]

[1] Visual Modelling Group, University of Brighton, UK
www.cmis.brighton.ac.uk/research/vmg
[2] University of Kent, Canterbury, UK
s.j.thompson@kent.ac.uk

Abstract. Spider diagrams are a visual notation for expressing logical statements. In this paper we identify a well known fragment of first order predicate logic, that we call $\mathcal{ESD}$, equivalent in expressive power to the spider diagram language. The language $\mathcal{ESD}$ is monadic and includes equality but has no constants or function symbols. To show this equivalence, in one direction, for each diagram we construct a sentence in $\mathcal{ESD}$ that expresses the same information. For the more challenging converse we show there exists a finite set of models for a sentence 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.

1 Introduction

Euler diagrams [2] exploit topological properties of enclosure, exclusion and intersection to represent subset, disjoint sets and set intersection respectively. Diagram d_1 in figure 1 is an Euler diagram and expresses that nothing is both a car and a van. Venn diagrams [13] are similar to Euler diagrams. In Venn diagrams, all possible intersections between contours must occur and shading is used to represent the empty set. Diagram d_2 in figure 1 is a Venn diagram and also expresses that no element is both a car and a van.

Many visual languages have emerged that extend Euler and Venn diagrams. One such language is Venn-II introduced by Shin [9]. Diagram d_3 in figure 1 is a Venn-II diagram. In addition to what is expressed by the underlying Venn diagram, it also expresses, using an *x-sequence*, the set $Cars \cup Vans$ is not empty. Venn-II diagrams can express whether a set is empty or not empty. Shin [9] shows that Venn-II is equivalent in expressive power to a first order language that she calls $\mathcal{L}_0$. The language $\mathcal{L}_0$ is a pure monadic language (i.e. all the predicate symbols are 'one place') that does not include constants or function symbols.

Another visual language, called Euler/Venn, based on Euler diagrams is discussed in [12].

These diagrams are similar to Venn-II diagrams but, instead of x-sequences, *constant sequences* are used. Diagram d_4 in figure 2 is an Euler/Venn diagram and expresses that no element is both a car and a van and that there is something called 'ford' that is either a car or a van. In [12] Swoboda and Allwein give an algorithm that determines whether a given monadic first order formula is

A. Blackwell et al. (Eds.): Diagrams 2004, LNAI 2980, pp. 112–127, 2004.

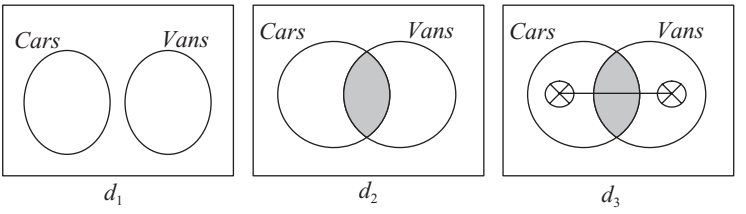

Fig. 1. An Euler diagram, Venn diagram and a Venn-II diagram

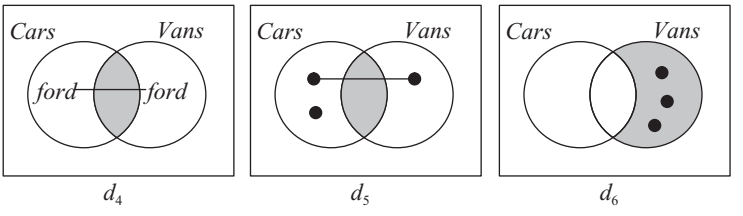

Fig. 2. An Euler/Venn diagram and two spider diagrams

observable from a given diagram. If the formula is observable from the diagram then it may contain weaker information than the diagram (i.e. the formula is a consequence of the information contained in the diagram).

Like Euler/Venn diagrams, spider diagrams are based on Euler diagrams. Rather than allowing the use of constant sequences[1] as in Euler/Venn diagrams, *spiders* denote the existence of elements. The spider diagram d_5 in figure 2 expresses that no element is both a car and a van and there are at least two elements, one is a car and the other is a car or a van. The spider diagram d_6 expresses that there are exactly three vans that are not cars. By allowing lower and upper bounds (by the use of shading and spiders) to be placed on the cardinality of sets, spider diagrams increase expressiveness over Venn-II.

We show, but do not include any proofs, that the spider diagram language is equivalent in expressive power to a fragment of first order logic that we call $\mathcal{ESD}$ (for the Expressiveness of Spider Diagrams). The language $\mathcal{ESD}$ extends $\mathcal{L}_0$ by adding equality, so $\mathcal{ESD}$ is monadic predicate logic with equality.

In section 5, we address the task of mapping each diagram to a sentence expressing the same information, showing that spider diagrams are at most as expressive as $\mathcal{ESD}$. In section 6 we show that $\mathcal{ESD}$ is at most as expressive as spider diagrams. We will outline Shin's algorithmic approach to show $\mathcal{L}_0$ (in which there is no equality) is not more expressive than Venn-II. It is simple to adapt this algorithm to find a spider diagram that expresses the same informa-

[1] In some spider diagram languages, **given spiders** [5] represent constants but for our purposes spiders represent existential quantification.

tion as a sentence in $\mathcal{ESD}$ that does not involve equality. However, for sentences in $\mathcal{ESD}$ that do involve equality, the algorithm does not readily generalize.

Thus, the task of showing that there exists a diagram expressing the same information as a sentence involving equality is challenging and we take a different approach. To motivate our approach we consider relationships between models for diagrams. We consider the models for a sentence and show that there is a finite set of models that can be used to classify all the models for the sentence. These classifying models can then be used to construct a diagram that expresses the same information as the sentence.

2 Spider Diagrams

In diagrammatic systems, there are two levels of syntax: concrete (or token) syntax and abstract (or type) syntax [4]. Concrete syntax captures the physical representation of a diagram. Abstract syntax 'forgets' semantically unimportant spatial relations between syntactic elements in a concrete diagram. We include the concrete syntax to aid intuition but we work at the abstract level.

2.1 Informal Concrete Syntax

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 the bounded area of 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. A **spider** is a tree with nodes (called **feet**) placed in different zones. The connecting edges (called **legs**) are straight lines. A spider **touches** a zone if one of its feet appears in that region. A spider is said to **inhabit** the region which is the union of the zones it touches. This union is called the **habitat** of the spider.

A **concrete unitary (spider) diagram** is a single boundary rectangle together with a finite collection of contours, shading and spiders. No two contours in the same unitary diagram can have the same label.

Example 1. Spider diagram d_6 in figure 2 has two contours and four zones. The shaded zone contains three spiders, each with one foot.

2.2 Formal Abstract Syntax

We can think of the contour labels used in our diagrams as being chosen from a countably infinite set, $\mathcal{L}$.

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.*

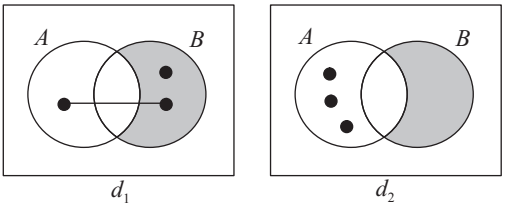

Fig. 3.

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.

When we reason with a spider diagram, the contour set may change, which is why we define an abstract zone to be a pair. Zone (a, b) is included in a but not included in b. Every contour in a concrete diagram contains at least one zone, captured by condition 2 (i). In any concrete diagram, the zone inside the boundary rectangle but outside all the contours is present, captured by condition 2 (ii). In order to give a unique abstraction from a concrete diagram we use spider identifiers (essentially a bag of spiders) rather than an arbitrary set of spiders.

Example 2. Diagram d_1 in figure 3 has abstract description

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 $\{(1, \{(\{B\}, \{A\})\}), (1, \{(\{A\}, \{B\}), (\{B\}, \{A\})\})\}$.

We define, for unitary diagram d, the **Venn zone set** to be $VZ(d) = \{(a, b) : a \subseteq L(d) \wedge b = L(d) - a\}$. If $Z(d) = VZ(d)$ then d is said to be in **Venn form**. If $z \in MZ(d) = VZ(d) - Z(d)$ then z is **missing** from d. Spiders represent the existence of elements and regions (an abstract region is a set of zones) represent sets – thus we need to know how many elements we have represented in each region. The number of spiders inhabiting region r_1 in d is denoted by $S(r_1, d)$. The number of spiders touching r_1 in d is denoted by $T(r_1, d)$, for more details see [6]. In d_1, figure 3, $(\{B\}, \{A\})$ is inhabited by one spider and touched by two spiders.

Unitary diagrams form the building blocks of compound diagrams. 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"). Some diagrams are not satisfiable and we introduce the symbol $\bot$, defined to be a unitary diagram interpreted as false. Our convention will be to denote unitary diagrams by d and arbitrary diagrams by D.

2.3 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 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 (regions are sets of zones). We define $\mathcal{Z}$ and $\mathcal{R}$ to be the sets of all abstract zones and regions respectively.

Definition 2. *An **interpretation of contour labels, zones and regions**, or simply an **interpretation**, is a pair (U, Ψ) 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. If $D = \bot$ then the **semantics predicate**, $P_D(m)$ is $\bot$. If D $(\neq \bot)$ is a unitary diagram then the **semantics predicate**, $P_D(m)$, of D 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) **Missing Zones Condition** Any zone, z, in $MZ(d)$ satisfies $\Psi(z) = \emptyset$.*

*If $D = D_1 \sqcup D_2$ then the **semantics predicate**, $P_D(m)$, of D is $P_D(m) = P_{D_1}(m) \vee P_{D_2}(m)$. If $D = D_1 \sqcap D_2$ then the **semantics predicate**, $P_D(m)$, of D is $P_D(m) = P_{D_1}(m) \wedge P_{D_2}(m)$. 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 3. Interpretation $m = (\{1, 2\}, \Psi)$ partially defined by $\Psi(A) = \{1\}$ and $\Psi(B) = \{2\}$ is a model for d_1 in figure 3 but not for d_2.

3 The Language $\mathcal{ESD}$

Spider diagrams can express statements of the form 'there are at least n elements in A' and 'there are at most m elements in A'. A first order language equivalent in expressive power to the spider diagram language will involve equality, to allow us to express the distinctness of elements, and monadic predicates, to allow us to express $x \in A$. In order to define such a language we require a countably infinite set of monadic predicate symbols, $\mathcal{P}$, from which all monadic predicate symbols will be drawn.

Definition 4. *The first order language $\mathcal{ESD}$ (for Expressiveness of Spider Diagrams) consists of the following.*

1. ***Variables**, $x_1, x_2, ...$ of which there are countably many.*
2. ***Atomic formulae**,*
   (a) if x_i and x_j are variables then $(x_i = x_j)$ is an atomic formula,
   (b) if $P_i \in \mathcal{P}$ and x_j is a variable then $P_i(x_j)$ is an atomic formula.
3. ***Formulae**, which are defined inductively.*
   (a) Atomic formulae are formulae.
   (b) $\bot$ and $\top$ are formulae.
   (c) If p and q are formulae so are $(p \wedge q)$, $(p \vee q)$ and $\neg p$.
   (d) If p is a formula and x is a variable then $(\forall x\, p)$ and $(\exists x\, p)$ are formulae.

We define $\mathcal{VAR}$, $\mathcal{F}$ and $\mathcal{S}$ to be the sets of variables, formulae and sentences (formulae with no free variables) of the language $\mathcal{ESD}$ respectively.

We shall assume the standard first order predicate logic semantic interpretation of formulae in this language, with the exception of allowing a structure to have an empty domain.

4 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 we shall assume, without loss of generality, that $\mathcal{L} = \{L_1, L_2, ...\}$ and $\mathcal{P} = \{P_1, P_2, ...\}$. Define $\mathcal{U}$ to be the class of all sets. The sets in $\mathcal{U}$ form the domains of structures in the language $\mathcal{ESD}$.

Definition 5. *Define $\mathcal{INT}$ to be the class of all interpretations for spider diagrams, that is*

$$\mathcal{INT} = \{(U, \Psi) : U \in \mathcal{U} \wedge \Psi \colon \mathcal{L} \cup \mathcal{Z} \cup \mathcal{R} \to \mathbb{P}U\}.$$

Define also $\mathcal{STR}$ to be the class of structures for the language $\mathcal{ESD}$, 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 $diag(U \times U)$.

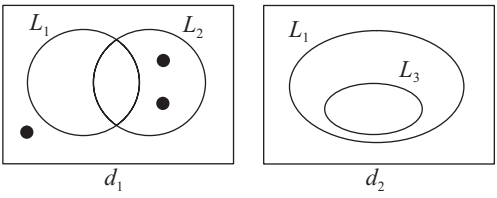

Fig. 4. Two α-diagrams: from diagrams to sentences

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 6. *Let D be a diagram and S be a sentence. We say D and S are* ***expressively equivalent*** *if and only if h provides a bijective correspondence between their models, that is*

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

5 Mapping from Diagrams to Sentences

To show that the spider diagram language is not more expressive than $\mathcal{ESD}$, we will map diagrams to expressively equivalent sentences. An ***α*-diagram** is a spider diagram in which all spiders inhabit exactly one zone [8]. Such diagrams have nice properties, for example, unitary α-diagrams only contain conjunctive information (spider legs represent disjunctive information).

Theorem 1. *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 (i.e. D_1 and D_2 have the same models).*

We will map each unitary α-diagram to an expressively equivalent sentence in $\mathcal{ESD}$. This enables us to map each disjunction of unitary α-diagrams to an expressively equivalent sentence and, by theorem 1, this is sufficient to show that the spider diagram language is not more expressive than the language $\mathcal{ESD}$.

Example 4. In diagram d_1, figure 4, there are three spiders, one outside both L_1 and L_2, the other two inside L_2 and outside L_1. 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

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

The disjunction of these sentences is expressively equivalent to $d_1 \sqcup d_2$. For general d_1 and d_2, the disjunction of their expressively equivalent sentences is expressively equivalent to $d_1 \sqcup d_2$.

To construct sentences for diagrams, it is useful to map zones to formulae.

Definition 7. *Define function* $\mathcal{ZOF}\colon \mathcal{Z} \times \mathcal{VAR} \to \mathcal{F}$ *(*$\mathcal{ZOF}$ *for 'zone formula') by, for each* $(a, b) \in \mathcal{Z} - \{(\emptyset, \emptyset)\}$ *and variable* x_j*,*

$$\mathcal{ZOF}((a, b), x_j) = \bigwedge_{L_k \in a} P_k(x_j) \wedge \bigwedge_{L_k \in b} \neg P_k(x_j)$$

and $\mathcal{ZOF}((\emptyset, \emptyset), x_j) = \top$.

We use the function $\mathcal{ZOF}$ to construct a sentence of $\mathcal{ESD}$ for each zone in a unitary α-diagram. We shall take these *zone sentences* in conjunction to give a sentence for the diagram. We define $\mathcal{D}_0^\alpha$ to be the class of all unitary α-diagrams and $\mathcal{D}^\alpha$ to be the class of all disjunctions of unitary α-diagrams.

Definition 8. *The partial function* $\mathcal{ZS}\colon \mathcal{Z} \times \mathcal{D}_0^\alpha \to \mathcal{S}$ *(*$\mathcal{ZS}$ *for 'zone sentence') is specified for unitary* α*-diagram* d *and zone* z *in* $VZ(d)$ *as follows.*

1. *If* z *is not shaded and not inhabited by any spiders then* $\mathcal{ZS}(z, d) = \top$.
2. *If* z *is not shaded and inhabited by* $n > 0$ *spiders then*

$$\mathcal{ZS}(z, d) = \exists x_1 ... \exists x_n (\bigwedge_{1 \leq j < k \leq n} \neg(x_k = x_j) \wedge \bigwedge_{1 \leq k \leq n} \mathcal{ZOF}(z, x_k)).$$

3. *If* z *is shaded or missing and not inhabited by any spiders then*

$$\mathcal{ZS}(z, d) = \forall x_1 \neg \mathcal{ZOF}(z, x_1).$$

4. *If* z *is shaded and inhabited by* $n > 0$ *spiders then*

$$\mathcal{ZS}(z, d) = \exists x_1 ... \exists x_n (\bigwedge_{1 \leq j < k \leq n} \neg(x_k = x_j) \wedge \bigwedge_{1 \leq k \leq n} \mathcal{ZOF}(z, x_k) \wedge \forall x_{n+1} (\bigvee_{1 \leq j \leq n} x_{n+1} = x_j \vee \neg \mathcal{ZOF}(z, x_{n+1}))).$$

Definition 9. *Define* $\mathcal{DS}\colon \mathcal{D}^\alpha \to \mathcal{S}$ *(*$\mathcal{DS}$ *for 'diagram sentence') as follows. Let* D *be a disjunction of unitary* α*-diagrams.*

1. *If* $D = \perp$ *then* $\mathcal{DS}(D) = \perp$.
2. *If* D $(\neq \perp)$ *is a unitary* α*-diagram then* $\mathcal{DS}(D) = \bigwedge_{z \in VZ(D)} (\mathcal{ZS}(z, D))$.
3. *If* $D = D_1 \sqcup D_2$ *then* $\mathcal{DS}(D) = (\mathcal{DS}(D_1) \vee \mathcal{DS}(D_2))$.

Theorem 2. *Let* D *be a disjunction of unitary* α*-diagrams. Then* D *is expressively equivalent to* $\mathcal{DS}(D)$.

Hence the language of spider diagrams is at most as expressive as $\mathcal{ESD}$.

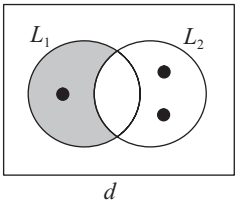

Fig. 5. Extending models for a diagram

6 Mapping from Sentences to Diagrams

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

Shin's approach for Venn-II and her language $\mathcal{L}_0$ ($\mathcal{ESD}$ without equality) is algorithmic [9], which we now outline. 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. A diagram can then be drawn for each of the simple parts of the resulting formula. To adapt this algorithm to sentences in $\mathcal{ESD}$ that do not involve equality is straightforward.

This algorithm does not readily generalize to arbitrary sentences in $\mathcal{ESD}$ 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 [1], 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 consider relationships between models for α-diagrams. We begin be considering a particular example.

Example 5. The diagram in figure 5 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 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 10. *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 $\mathcal{ESD}$, we will identify all 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 $\forall x_1\, P_1(x_1) \wedge \forall x_1\, \exists x_2\, x_1 \neq x_2$. 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 more than two elements in certain predicate intersection sets then S cannot place an upper bound on the cardinalities of these predicate intersection sets.

The interpretation of P_1 has to have all the elements, of which there must be 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 cannot place an upper bound on $|PI(m, \{P_1\}, \emptyset)|$. We can think of m_1 and m_2 extending m_3 with domain $U_3 = \{1, 2\}$ where $P_1^{m_3} = \{1, 2\}$ and $P_j = \emptyset$, for all $j \neq 1$.

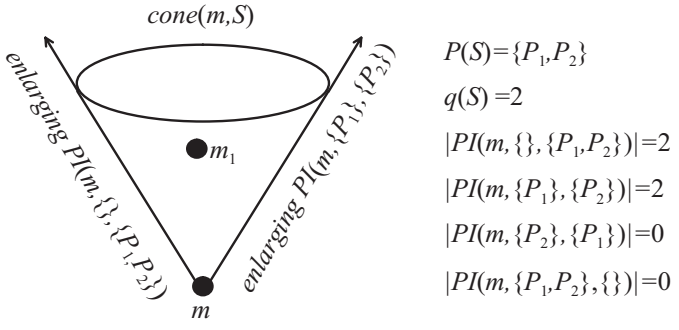

Fig. 6. Visualizing cones

Definition 11. *Let S be a sentence and define $q(S)$ to be the maximum number of nested quantifiers in S and $P(S)$ 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 [1].

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

Lemma 1. *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, [1].*

Lemma 1 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 $2^{|P(S)|}q(s)$. If the cardinality of model m for sentence S 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 12. *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 $|PI(m_1, X, P(S) - X)| < q(s)$.*

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' (keeping predicate intersection sets the same) m in others.

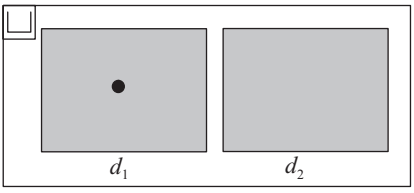

Fig. 7. A diagram expressively equivalent to $\forall x \forall y\, x = y$

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 figure 6. 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 $PI(m, \{P_1\}, \{P_2\}) \rightarrow PI(m_3, \{P_1\}, \{P_2\})$.

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 figure 7. 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.

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 13. *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 $\gamma : U_1 \rightarrow U_2$ such that*

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

*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 14. *Let S be a sentence. A set of small models, $class(S)$, 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 $class(S)$ such that m_1 and m_2 are isomorphic, restricted to $P(S)$.*

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

Definition 15. *Let m be a small model for sentence S. The unitary α-diagram, d,* ***representing*** *m, is defined as follows* [2].

1. *The contour labels arise from the predicate symbols in $P(S)$:*
$$\{L_i \in \mathcal{L} : \exists P_i \in \mathcal{P} \bullet P_i \in P(S)\}.$$
2. *The diagram is in Venn form:*
$$Z(d) = \{(a, b) : a \subseteq L(d) \wedge b = L(d) - a\}.$$
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:*

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

We write $\mathcal{REP}(m_1, S) = d$. Let $class(S)$ be a set of classifying models for S. Define $\mathcal{D}(S)$ to be a disjunction of unitary diagrams, given by

$$\mathcal{D}(S) = \bigsqcup_{m \in class(S)} \mathcal{REP}(m, S),$$

unless $class(S) = \emptyset$, in which case $\mathcal{D}(S) = \perp$.

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. Four of these structures are models for S and are listed below.

[2] In fact, d is a β-diagram [8] (every zone is shaded or inhabited by at least one spider).

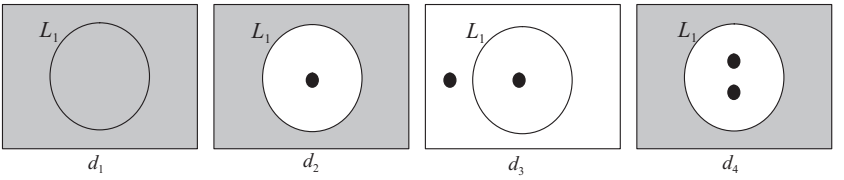

Fig. 8. Constructing diagrams from models

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 figure 8 respectively. 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 $class(S)$ be a set of classifying models for S. Then S is expressively equivalent to $\mathcal{D}(S)$.*

Hence the language of spider diagrams and $\mathcal{ESD}$ are equally expressive.

7 Conclusion

In this paper we have identified a well known fragment of first order predicate logic equivalent in expressive power to the spider diagram language. To show that the spider diagram language is at most as expressive as $\mathcal{ESD}$, we identified a sentence in $\mathcal{ESD}$ that expressed the same information as a given diagram. To show that $\mathcal{ESD}$ is at most as expressive as the language of spider diagrams we considered relationships between models for sentences. We have shown that it is possible to classify all the models for a sentence by a finite set of models. These models can be used to define a spider diagram expressively equivalent to S.

The spider diagram language extends to the far more expressive constraint diagram language [7]. Constraint diagrams allow relational navigation (expressions involving two place predicates). The diagram in figure 9 is a constraint diagram. In addition to the information provided by the underlying spider diagram, it expresses that 'for all x in $B - A$, the relational image of x under g is A and there is a y in $A - B$ whose relational image under f is an element of C'. It is currently unknown what fragment of first order predicate logic can be expressed using constraint diagrams. Various constraint diagram languages exist. The simplest of these restricts the syntactic components and the semantic

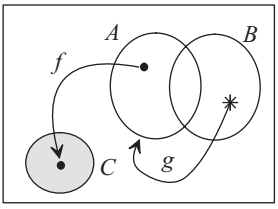

Fig. 9. A constraint diagram

interpretation of the diagrams [10]. In [3] the authors give a reading algorithm for interpreting more expressive constraint diagrams.

Some logical assertions are more naturally expressed in one language than another. This may lead to the development of heterogeneous reasoning systems. An example of such a system based on first order predicate logic and Euler/Venn diagrams can be found in [11]. We plan to develop a heterogeneous reasoning system incorporating constraint diagrams. The other languages included may be influenced by the expressiveness of the languages involved. Thus it will be useful to know how expressive constraint diagrams are. This work on the expressiveness of spider diagrams will lay the foundations for an investigation into the expressiveness of constraint diagrams.

Acknowledgment

Gem Stapleton thanks the UK EPSRC for support under grant number 01800274. John Howse, John Taylor and Simon Thompson are partially supported by the UK EPSRC grant numbers GR/R63516 and GR/R63509 for the reasoning with diagrams project. Thanks also to Andrew Fish and Jean Flower for their comments on earlier drafts of this paper.

References

[1] B. Dreben and D. Goldforb. *The Decision Problem. Solvable Classes of Quantificational Formulas*. Addison Wesley Publishing Company Inc., 1979. 120, 122

[2] L. Euler. Lettres a une princesse d'allemagne, 1761. vol 2, Letters No. 102-108. 112

[3] A. Fish, J. Flower, and J. Howse. A reading algorithm for constraint diagrams. In *Proceedings of* IEEE Symposium on Visual Languages and Formal Methods, pages 161–168. IEEE, 2003. 126

[4] J. Howse, F. Molina, S-J. Shin, and J. Taylor. On diagram tokens and types. In *Proceedings of Diagrams 2002*, LNCS, pages 76–90. Springer-Verlag, April 2002. 114

[5] 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. 113

[6] J. Howse, G. Stapleton, and J. Taylor. Spider diagrams. In available from `www.cmis.brighton.ac.uk/research/vmgSDRules`., 2003. 115
[7] S. Kent. Constraint diagrams: Visualising invariants in object oriented models. In *Proceedings of OOPSLA97, ACM SIGPLAN Notices*, 1997. 125
[8] F. Molina. *Reasoning with extended Venn-Peirce diagrammatic systems*. PhD thesis, University of Brighton, 2001. 118, 124
[9] S.-J. Shin. *The Logical Status of Diagrams*. Cambridge University Press, 1994. 112, 120
[10] 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 Institute, 2003. 126
[11] N. Swoboda and G. Allwein. A case study of the design and implementation of heterogeneous reasoning systems. In *Logical and Computational Aspects of Model-Based Reasoning*. Kluwer Academic, 2002. 126
[12] N. Swoboda and G. Allwein. Using DAG transformations to verify Euler/Venn homogeneous and Euler/Venn FOL heterogeneous rules of interence. In *International Workshop on Graph Transformation and Visual Modeling Techniques*, pages 84–98, Barcelona, October 2002. 112
[13] J. Venn. On the diagrammatic and mechanical representation of propositions and reasonings. *The London, Edinburugh and Dublin Philosophical Magazine and Journal of Science*, 9:1–18, 1880. 112