Spider Diagrams Augmented with Constants: A Complete System

Gem Stapleton
Visual Modelling Group
University of Brighton, UK
g.e.stapleton@brighton.ac.uk

Abstract

The use of visual languages in computing is varied, ranging from system modelling to the display of data analyzed in computation processes. A prominent example of a visual notation is the Unified Modelling Language (UML), designed for use by software engineers. Constraint diagrams were proposed as an alternative to the UML's Object Constraint Language. Spider diagrams form a fragment of constraint diagrams, and their applications are more broad than just placing constraints on software models, including information visualization and displaying the results of database queries. This paper focuses on spider diagrams augmented with constants that represent specific individuals. We present a sound reasoning system for spider diagrams with constants and establish completeness. Furthermore, the technique used to prove completeness can be adapted to give rise to a decision procedure.

1 Introduction

It is widely recognized that diagrams play an important role in various areas particularly in many aspects of computing, including visualizing information and reasoning about that information. Diagrams are often useful for conveying complex information in accessible and intuitive ways. This is one reason behind the widening perception of the importance of diagrams in computing systems and more widely.

Software engineers form one group of users that need formal languages to specify and design complex systems. Ideally, their software specifications should be accessible to all stakeholders involved in the modelling process, including customers, managers and programmers. There is extensive use of diagrams to model software, with the Unified Modelling Language (UML) being an industry standard, mainly visual, notation. The only non-diagrammatic part of the UML is the Object Constraint Language (OCL) which is designed to place formal constraints on software models. It, therefore, seems sensible to offer formal diagrammatic notations for the purpose of precise, yet accessible, software specification.

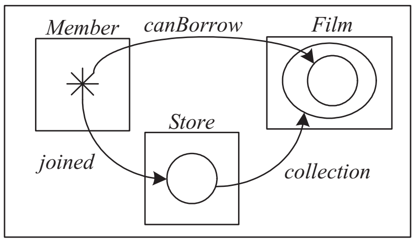

Figure 1. A constraint diagram.

Constraint diagrams were introduced in [10] as a way to visualize object-oriented invariants in the context of the UML. They have been used to develop high-level models independently of UML [8, 12]. Building on Euler and Venn diagrams, constraint diagrams contain *spiders* to indicate existential and universal quantification and use arrows to make statements about binary relations. For example, the constraint diagram in figure 1 expresses that people can borrow only books that are in the collections of libraries that they have joined. A formalization of constraint diagrams can be found in [4] and a generalized version of them is formalized in [16].

The language of spider diagrams [9] forms a fragment of the constraint diagram language. The only spiders present in spider diagrams in [9] represent the existence of elements (called *existential spiders*) and arrows are not permitted. The spider diagram d_1 in figure 2 expresses, by the inclusion of the curve $Lions$ inside $Cats$, that all lions are cats and, in addition, it expresses that there is a cat, which may or may not be a lion by the use of the existential spider (i.e. the tree). The spider diagram d_2 expresses is something which is not a dog. It has been shown that the spider diagram language is equivalent in expressive power to monadic first order logic with equality [18].

The diagrams d_4, d_5 and d_6 include constant spiders; these are labelled trees whose nodes are visualized as squares. The diagram d_4 tells us that tom is a cat; d_5 tells

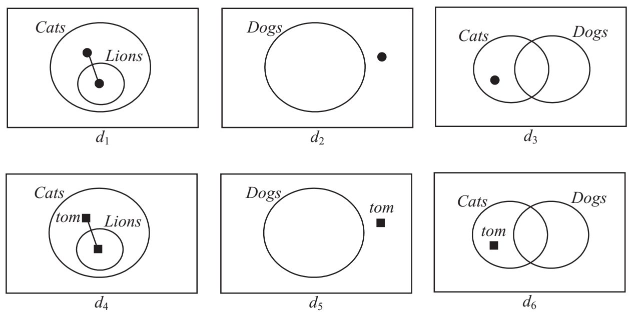

Figure 2. Spider diagrams.

us that tom is not a dog. From d_4 and d_5 we can deduce d_6 which tells us that tom is a cat but not a dog. By contrast, from d_1 and d_2 we cannot deduce d_3 (something is a cat but not a dog). There are many notations related to spider and constraint diagrams; see [1, 6, 15, 19] for examples.

There are a number of examples of spider diagrams being used in practice, such as assisting with the task of identifying component failures in safety critical hardware designs [2] and in other domains such as [14]. They have also been used (but not explicitly) for displaying the results of database queries [20], representing non-hierarchical computer file systems [3], in a visual semantic web editing environment [13, 21] and other areas [7, 11]. Virtually all of these application areas, constants are used to represent specific objects, thus highlighting the importance of augmenting spider diagrams with constants.

The contribution made in this paper is to provide a set of sound and complete reasoning rules for spider diagrams augmented with constants. Section 2 overviews the syntax and semantics of spider diagrams with constants. A set of reasoning rules is presented in section 3. Soundness and completeness is established in section 4; the proof strategies are only sketched due to space limits. The technique used to prove completeness can be trivially adapted to provide a decision procedure for spider diagrams with constants.

2 Syntax and Semantics

We give an informal overview of the syntax and semantics of spider diagrams with constants; a formalization can be found in [17]. The spider diagram d_1 in figure 2 contains two labelled, closed curves called *contours*. The minimal regions that can be described as inside certain (possibly no) contours and outside the remaining contours are called *zones*; d_1 contains three zones whereas d_2 contains just two zones. *Missing zones* are zones which could be present in a diagram, given the contour label set, but are not present; for example, in figure 2, d_1 has exactly one missing zone: that which is inside $Lions$ but outside $Cats$. The diagram

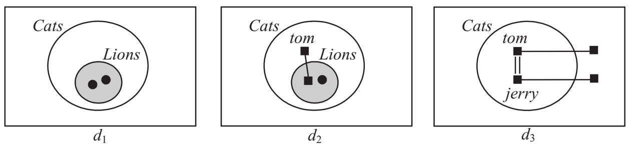

Figure 3. Shading and ties.

d_2 does not have any missing zones.

Spiders are placed in *regions*; a region is a set of zones. The region in which a spider is placed is called its *habitat*. Visually, a spider is represented by a tree whose nodes are either all round or all square; these nodes are called *feet*. Spiders with round feet are *existential* whereas those with square feet are *constant*. Constant spiders are labelled. The contours and spiders are all contained by a boundary rectangle which illustrates 'where the diagram stops'. Diagrams enclosed by such a rectangle are called *unitary diagrams*. In a unitary diagram, no two labels appear twice. Moreover, the labels used for constant spiders are never used for contour labels. This applies globally, in that any constant spider label never labels a contour and vice versa.

Unitary diagrams also contain further syntax: *shading* and *ties*. Shading is placed in zones, as demonstrated in figure 3. Ties can be placed between any pair of constant spider feet that are placed in the same zone. We treat the relation of two feet being joined by a tie as transitive: given any constant spider feet f_1, f_2 and f_3, if f_1 is joined to f_2 by a tie and f_2 is joined to f_3 by a tie then f_1 is joined to f_3. Moreover, we also assume that each foot is joined by a tie to itself (i.e. the relation is reflexive). This simplifies the formalization of the semantics; see [17]. Visually, ties are a pair of parallel line segments, like an equals sign, also demonstrated in d_3, figure 3. Note that, rather than drawing all ties between feet, we draw essentially a spanning forest of the graph that represents the 'is joined to be a tie' relation. Given two constant spiders, their *web* is the set of zones in which their feet are joined by a tie; in d_3, figure 3, the web of tom and $jerry$ is the single zone inside $Cats$.

We note that ties could also be used to connect existential spider feet. Indeed, they could also be used to connect an existential foot to a constant foot. However, for any diagram that incorporated such ties there exists a semantically equivalent diagram that does not contain such ties. This is not the case for ties between constant spider feet. It is straightforward to extend the work in this paper to the case where these additional ties are permitted.

In addition to the above, we take the symbol $\perp$ to be a unitary diagram. Unitary diagrams form the building blocks of *compound diagrams*: if d_1 and d_2 are spider diagrams then so are $(d_1 \vee d_2)$ and $(d_1 \wedge d_2)$.

For the semantics, regions represent sets, as illustrated in the introduction. Spiders represent elements in the sets represented by their habitats and distinct spiders represent distinct elements unless they are joined by a tie. Constant spiders represent specific individuals, just like contours represent specific sets; these individuals and sets are determined, in part, by their labels. Slightly more formally, an *interpretation* consists of a universal set, U, and a mapping of contour labels to subsets of U and constant spider labels to elements of U. The mapping of contour labels extends to the interpretation of zones and regions; see [17]. Two constant spiders represent the same individual if and only if they both represent an individual which is in the set denoted by some zone in their web. Shading places an upper bound on set cardinality: in the set represented by a shaded zone, all of the elements must be represented by spiders.

In figure 3, d_1 asserts that all lions are cats, there are at least two lions (by the use of two existential spiders) and there are no more than two lions (by the use of shading); in other words, all lions are cats and there are exactly two lions. The diagram d_2 expresses that all lions are cats, tom is a lion or a cat, something else is a lion and there are at most two lions. The diagram d_3 uses a tie to indicate that tom and $jerry$ are the same individual whenever they are both cats whereas if at least one of them is not a cat, they do not denote the same individual (due to the absence of a tie between their feet in the zone outside $Cats$).

The diagram $\perp$ is interpreted as false. The semantics extend to compound diagrams in the obvious way. Informally, we say that an interpretation is a *model* for a diagram when it 'agrees with the meaning of the diagram' as described above. Diagrams which have models are said to be *satisfiable*. The following theorem tells us that unitary diagrams are not capable of expressing contradictory information.

Theorem 2.1 *Let* d $(\neq\perp)$ *be a unitary spider diagram with constants. Then* d *is satisfiable.*

Given diagrams d_1 and d_2, we say that d_1 **semantically entails** d_2, denoted $d_1 \vDash d_2$, if all of the models for d_1 are also models for d_2.

3 Reasoning Rules

We will now develop a set of sound and complete reasoning rules for spider diagrams with constants. All of the reasoning rules given for spider diagrams without constants in [9] can be modified (sometimes non-trivially) and extended to spider diagrams with constants. In addition, new rules are also required.

3.1 Unitary to unitary reasoning rules

We introduce a collection of reasoning rules that apply to, and result in, a unitary diagram. For example, we can delete shading from a unitary diagram. All of the rules in this section are extended from those for spider diagrams without constants. The first rule allows the introduction of a shaded zone to a unitary diagram, d.

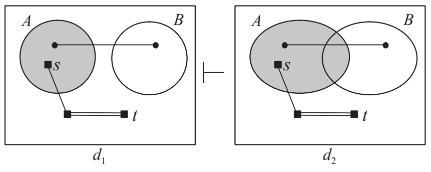

Figure 4. Introducing a shaded zone.

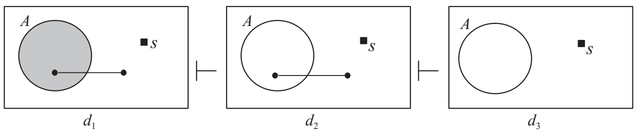

Figure 5. Erasing shading and spiders.

Rule 1 ***Introduction of a shaded zone*** Let d_1 be a unitary diagram that has a missing zone. If d_2 is a copy of d_1 except that d_2 contains a new, shaded zone then d_1 may be replaced by d_2 and vice versa.

In figure 4, rule 1 (introduction of a shaded zone) is applied to d_1 to give d_2. Applying the introduction of a shaded zone rule results in a semantically equivalent diagram. The next two rules are not information preserving.

Rule 2 ***Erasure of shading*** Let d_1 be a unitary diagram with a shaded region r. Let d_2 be a copy of d_1 except that r is completely non-shaded in d_2. Then d_1 may be replaced by d_2.

In figure 5, rule 2 (erasure of shading) is applied to d_1 to give d_2. Applying this rule 'forgets' the upper bound on the cardinality of the set represented by the region from which shading is erased.

Rule 3 ***Erasure of an existential spider.*** Let d_1 be a unitary diagram containing an existential spider e with a completely non-shaded habitat. Let d_2 be a copy of d_1 except that d_2 does not contain e. Then d_1 may be replaced by d_2.

In figure 5, rule 3 (erasure of an existential spider) is applied to d_2 to give d_3.

3.2 Unitary to compound reasoning rules

We now specify five further rules, each of which is reversible, that allow a unitary diagram to be replaced by a compound diagram, including a rule that allows us to introduce a contour. In the spider diagram without constants

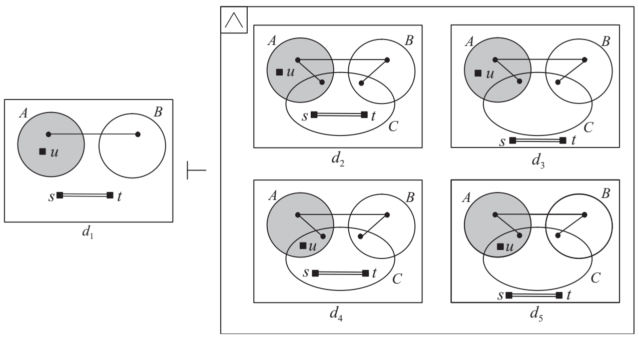

Figure 6. Diagram C-extensions.

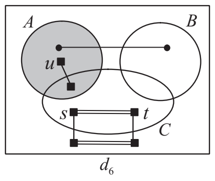

Figure 7. Incorrectly introducing a contour.

system, the introduction of a contour rule applies to, and results in, a unitary diagram. In figure 6, we can introduce the contour with label C to d_1. When we do so, each zone must split into two new zones, thus ensuring that information is preserved. The existential spiders' feet bifurcate, one new foot is placed in each new zone of the habitat. More care must be taken with the constant spiders, however. Consider, for example, the constant spiders s and t. The individual represented by both s and t must be either in $C - (A \cup B)$ or in $U - (A \cup B \cup C)$. The constant spider u represents an individual that is either in $A - (B \cup C)$ or $(A \cap C) - B$. This gives rise to four possibilities, shown in d_2, d_3, d_4 and d_5. We call these four diagrams the *C-extensions* of d_1. The diagram d_1 is semantically equivalent to $d_2 \vee d_3 \vee d_4 \vee d_5$. We could replace d_1 with the disjunction of just two unitary diagrams, each with u having two feet: one foot in the zone just in A, the other in the zone inside A and C.

It is not the case that the single unitary diagram d_6, in figure 7 is semantically equivalent to d_1. The constant spiders s and t must represent the same individual in d_1 but this is not the case in d_6, since the semantics of ties are zone based. To define the introduction of a contour rule, we first define the component parts of the resulting disjunction. We call these component parts *L-extensions*, where L is the introduced contour label.

Definition 3.1 *Let d_1 be a unitary diagram such that each constant spider in d_1 has exactly one foot. Let L be a contour label that is not in d_1. Let d_2 be a unitary diagram such that each constant spider in d_2 has exactly one foot. If the following conditions hold then d_2 is an **L-extension** of d_1.*

1. *The labels in d_2 are those in d_1, together with L.*
2. *The constant spider labels match.*
3. *The zones in d_2 are as follows:*
    (a) *each zone in d_1 is split into two zones in d_2, one inside and the other outside L;*
    (b) *shading is preserved in that if a zone is shaded in d_1 then the two zones it splits into in d_2 are both shaded in d_2 and no others are shaded in d_2.*
4. *The existential spiders match and their habitats are preserved under 'zone splitting'.*
5. *The habitat of each constant spider, c, in d_2 is either the zone inside L or that outside L (but not both) arising from its habitat it d_1.*
6. *Spider webs are preserved. Since constant spiders are single footed, this means that spiders joined by a tie in d_1 have the same habitats as each other in d_2.*

The set of L-extensions of d_1 is denoted $\mathcal{EXT}(L, d_1)$.

Rule 4 ***Introduction of a contour label*** Let d_1 $(\neq \bot)$ be a unitary diagram such that each constant spider has exactly one foot. Let L be a label not in d_1. Then d_1 may be replaced by $\bigvee\limits_{d_2 \in \mathcal{EXT}(L,d_1)} d_2$ and vice versa.

Rule 5 ***Splitting spiders*** Let d be a unitary diagram with a spider s touching precisely every zone of two disjoint regions r_1 and r_2. Let d_1 and d_2 be unitary diagrams that are copies of d except that neither contains s, but instead each contains an extra spider, s_1 and s_2 respectively, whose habitats are regions r_1 in d_1 and r_2 in d_2. If s is a constant spider, then

1. s_1 and s_2 have the same label as s and
2. any ties joined to any given foot of s in d are joined to the appropriate foot of s_1 in d_1 or s_2 in d_2.

Then d can be replaced by the diagram $d_1 \vee d_2$ and vice versa.

Figure 8 illustrates an application of the splitting spiders rule. The spider s in d splits into two spiders, one in d_1, the other in d_2. Intuitively, the individual represented by s is either in the set $U - (A \cup B)$ or the set $A \cup B$.

Rule 6 ***Excluded middle*** Let d be a unitary diagram with a completely non-shaded region r. Let d_1 and d_2 be unitary diagrams that are copies of d except that d_1 contains an extra existential spider whose habitat is r and, in d_2, r is shaded. Then d can be replaced by the diagram $d_1 \vee d_2$ and vice versa.

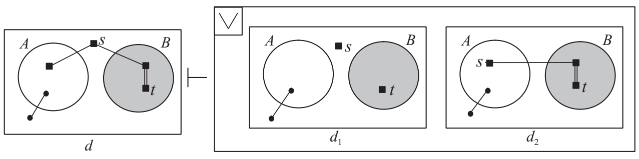

Figure 8. Splitting spiders.

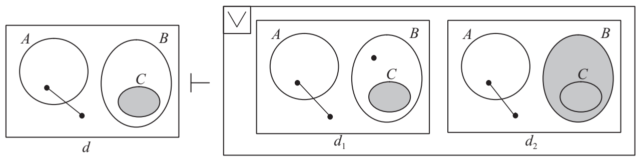

Figure 9. Excluded middle.

The diagram d in figure 9 can be replaced by $d_1 \vee d_2$ by applying the excluded middle rule.

Given a unitary diagram, d, that has only non-empty models (in which case d contains at least one spider), we can deduce that the individual represented by a constant spider, t, belongs to one of the sets denoted by the zones in d. Moreover, this individual must either be the same as, or different from, the elements already represented in d. As an example, consider d in figure 10 which has only non-empty models. Thus, in any model for d the constant spider t represents some individual. Then t is in A, B or $U - (A \cup B)$. If t is in A then it must equal s, since the region inside A is entirely shaded, shown in d_1. If t is in the set B then it may be either equal to, or different from, the element represented by the existential spider in B in the diagram d; these cases are represented by d_2 and d_3 respectively. Finally, if t is not in A or B then t must be in $U - (A \cup B)$, represented by d_4. The diagrams d_1, d_2, d_3 and d_4 are called *t-extensions* of d. For simplicity, we only add a constant spider to a diagram when all present spiders have exactly one foot; such a diagram is called an **α-diagram**.

Definition 3.2 *Let d_1 be a unitary α-diagram containing at least one spider and let t be some constant spider (label) that is not in d_1. Let d_2 be a unitary α-diagram. If the only differences between d_1 and d_2 are the following then d_2 is a **t-extension** of d_1:*

1. *d_2 contains t with an arbitrary single zone habitat,*
2. *if the habitat of t is shaded then it is joined to another constant spider by a tie or there is one fewer existential spider in that zone (but not both),*
3. *if the habitat, z, of t is not shaded then either*
   - *(a) the number of existential spiders inhabiting z is reduced by one, or*
   - *(b) the number of existential spiders inhabiting z is the same, or*
   - *(c) t is joined by a tie to some constant spider also inhabiting z and the number of existential spiders is the same.*

The set of all t-extensions of d_1 is denoted $\mathcal{EXT}(t, d_1)$.

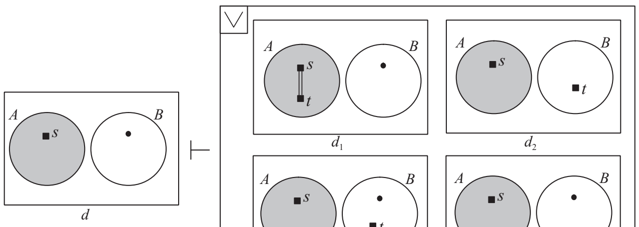

Figure 10. Diagram t-extensions.

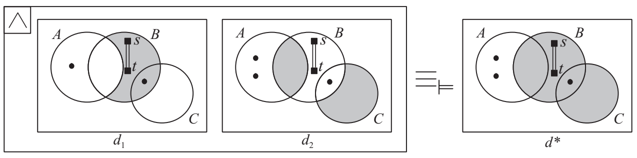

Figure 11. Combining diagrams.

Rule 7 ***Introduction of a constant spider*** Let d_1 be a unitary α-diagram that contains at least one spider and let t be a constant spider not in d_1. Then d_1 can be replaced by the diagram $\bigvee_{d_2 \in \mathcal{EXT}(t,d_1)} d_2$ and vice versa.

Introducing the constant spider t to d in figure 10, by applying rule 7 results in $d_1 \vee d_2 \vee d_3 \vee d_4$.

The final rule in this section, called *combining*, replaces two unitary α-diagrams, with the same zone sets and constant spider label sets, taken in conjunction by a single unitary diagram. In figure 11, we illustrate the combining rule. We combine $d_1 \wedge d_2$ to give d^*. Any shading in either d_1 or d_2 occurs in d^*. Moreover, the number of spiders in any zone in d^* is the same as the maximum number that occur in that zone in d_1 or d_2. The diagram $d_1 \wedge d_2$ is semantically equivalent to d^*.

We now give a further example in a build-up to the definition of combining. In this example we derive results by working at the semantic level, but we will define the combining rule at the syntactic level. In figure 12, d_1 and d_2 contain contradictory information. We observe the following.

1. The zone inside A but outside B and C is shaded in d_1 and contains more spiders in d_2. Moreover, z_1 represents the empty set in any model for d_1. In any model for d_2, z_1 does not represent the empty set.

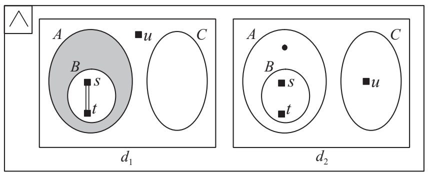

Figure 12. An unsatisfiable diagram.

2. The constant spider u has different habitats in the two diagrams. In any model for d_1, u represents an individual that is not in the set $A \cup C$. In any model for d_2, u represents an individual in the set C.

3. The constant spiders s and t are joined by a tie in d_1 but not in d_2. In any model for d_1, s and t represent the same individual, but in any model for d_2 they represent distinct individuals.

From any one of these three observations we can deduce that $d_1 \wedge d_2$ is unsatisfiable.

Definition 3.3 *Let d_1 and d_2 be unitary α-diagrams such that one of the following three conditions holds.*

1. The zones are the same and the constant spider labels are the same.

2. The zones are the same and d_1 or d_2 is entirely shaded.

3. $d_1 =\perp$ or $d_2 =\perp$.

*Then d_1 and d_2 are called **similar**.*

Definition 3.4 *Similar unitary α-diagrams d_1 and d_2 are said to be in **contradiction** if and only if one of the following four conditions holds.*

(i) Either $d_1 =\perp$ or $d_2 =\perp$.

(ii) There is a zone that is shaded in one diagram and contains more spiders in the other.

(iii) There is a constant spider with different habitats in d_1 and d_2.

(iv) There are two constant spiders that are joined by a tie in one diagram but not the other.

Lemma 3.1 *Let d_1 and d_2 be similar unitary α-diagrams.Then d_1 and d_2 are in contradiction if and only if $d_1 \wedge d_2$ is unsatisfiable.*

For completeness, it is sufficient to stipulate that the combining rule only applies to diagrams that have the same sets of constant spiders or that are contradictory.

Definition 3.5 *Let d_1 and d_2 be similar unitary α-diagrams Then their **combination**, denoted $d^* = d_1 * d_2$, is a unitary α-diagram defined as follows.*

1. If d_1 and d_2 are in contradiction then $d_1 * d_2 =\perp$.

2. Otherwise $d^* = d_1 * d_2$ is a unitary α-diagram such that the following hold.

(a) The set of zones in the combined diagram is the same as the set of zones in the original diagrams.

(b) The shaded zones in the combined diagram are shaded in one (or both) of the original diagrams.

(c) The number of existential spiders in any zone in the combined diagram is the maximum number of existential spiders inhabiting that zone in the original diagrams.

(d) The constant spiders in the combined diagram are the same as those in the original diagrams.

(e) The habitats of the constant spiders in the combined diagram are the same as those in the original diagrams.

(f) The webs of the constant spiders in the combined diagram are the same as those in the original diagrams.

Rule 8 ***Combining*** *Let d_1 and d_2 be similar unitary α-diagrams. Then $d_1 \wedge d_2$ may be replaced by $d_1 * d_2$, and vice versa.*

3.3 Logic reasoning rules

We now introduce a collection of rules, all of which have (obvious) analogies in logic. For space reasons, we give few details; throughout, D_1, D_2 and D_3 are arbitrary diagrams.

1. **Connect a diagram** D_1 can be replaced by $D_1 \vee D_2$.

2. **Inconsistency** $\perp$ can be replaced by D_1.

3. **$\vee$–Idempotency** D_1 may be replaced by $D_1 \vee D_1$ and vice versa.

4. **$\wedge$–Idempotency** D_1 may be replaced by $D_1 \wedge D_1$ and vice versa

We also assume that we have associativity, commutativity and distributivity.

3.4 Obtainability

To conclude this section on reasoning rules we define obtainability.

Definition 3.6 *Let D_1 and D_2 be two diagrams. Then D_2 is **obtainable** from D_1, denoted $D_1 \vdash D_2$, if and only if there is a sequence of diagrams $\langle D^1, D^2, ..., D^m \rangle$ such that $D^1 = D_1$, $D^m = D_2$ and D^k can be transformed into D^{k+1} by an application of a reasoning rule, for each k (where $1 \leq k < m$). If $D_1 \vdash D_2$ and $D_2 \vdash D_2$, we write $D_1 \equiv_\vdash D_2$.*

4 Soundness, Completeness and Decidability

To prove that the system is sound, the strategy is to start by showing that each reasoning rules is sound. The soundness theorem then follows by a simple induction argument. Due to space reasons, we omit the proofs.

Theorem 4.1 ***Soundness*** *Let D_1 and D_2 be spider diagrams. If $D_1 \vdash D_2$ then $D_1 \vDash D_2$.*

The completeness proof strategy for spider diagrams without constants given in [9] extends to the more general case here. The extended strategy, outlined in figure 13, is as follows. Suppose that $D_1 \vDash D_2$. The aim is to transform D_1 and D_2 into disjunctions of unitary α-diagrams using reversible rules, where, roughly speaking, each unitary part has some specified contour label set and constant spider label set.

Firstly, we split the constant spiders in D_1 and D_2 until, in each unitary part, all the constant spiders have precisely one foot, giving D_1^S and D_2^S respectively. This allows us to add contours to the unitary parts in both D_1^S and D_2^S using the reversible rule 4 (introduction of a contour label), until each (non-false) unitary part has the same contour label set, L. This gives D_1^L and D_2^L respectively. For the next step, zones are introduced to each unitary part until all (non-false) unitary parts have the same zone set, Z. This is done using the reversible rule 1 (introduction of a shaded zone) and yields D_1^Z and D_2^Z respectively. Now we obtain α-diagrams using the reversible rule 5 (splitting spiders), yielding D_1^α and D_2^α respectively. The formalization of the diagrams D_i^L, D_i^Z and D_i^α generalize those given in [9] for spider diagrams without constants.

We wish to introduce constant spiders to each side until each unitary part has the same constant spider label set. However, we can only introduce constant spiders when our diagrams contain at least one spider (ensuring non-empty models). Thus the next step we take is to apply the excluded middle rule to both sides until all the (non-false) unitary parts are either entirely shaded or contain a spider. The reversible rule 7 (introduction of a constant spider) is then applied, introducing constant spiders to all unitary parts that contain a spider, until all such unitary parts have some specified constant spider label set, C. This gives D_1^C and D_2^C respectively. We now apply rule 8 (combining) and some

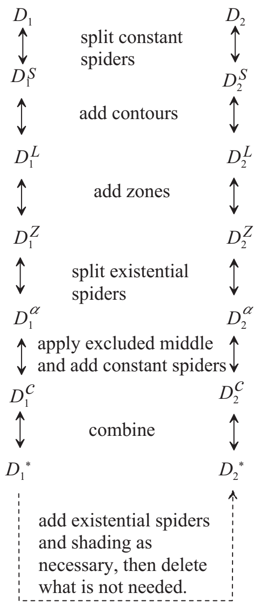

Figure 13. Proof strategy.

logic rules to remove all the conjuncts, giving two disjunctions of unitary α-diagrams, D_1^* and D_2^*. All of the unitary parts of D_1^* and D_2^* are either

1. $\perp$ or
2. have zone set Z and are entirely shaded and contain no spiders or
3. have zone set Z and constant spider label set C.

We note that $D_1 \equiv_\vdash D_1^*$ and $D_2 \equiv_\vdash D_2^*$, since all the rules applied so far are reversible. The diagram D_1^* (D_2^*) is a type of normal form that reflects the semantics of D_1 (D_2) in a clear manner.

We now apply the excluded middle rule to D_1^* until there are sufficiently many existential spiders and enough shading to ensure that each unitary part on the left hand side syntactically entails a unitary part of D_2^*, giving D_1'. Hence $D_1 \vdash D_1^* \vdash D_2^* \vdash D_2$ and, therefore, $D_1 \vdash D_2$.

The major differences between the completeness proof strategy here and that for spider diagrams without constants are the addition of the first step (splitting the constant spiders), with knock on changes to details of the other steps; and the insertion of an extra stage between splitting existential spiders and combining diagrams. Showing that $D_1^* \vdash D_2^*$ is also more challenging.

Theorem 4.2 ***Completeness*** *Let D_1 and D_2 be spider diagrams with constants. Then $D_1 \vDash D_2$ implies $D_1 \vdash D_2$.*

The proof of completeness provides an algorithmic method for constructing a proof that $D_1 \vdash D_2$ whenever

$D_1 \vDash D_2$. It is simple to adapt this algorithm to determine, given any D_1 and D_2, whether $D_1 \vdash D_2$.

Theorem 4.3 ***Decidability*** *Let D_1 and D_2 be constraint diagrams. There exists an algorithm that determines whether $D_1 \vdash D_2$.*

5 Conclusion

In this paper we have developed a sound and complete reasoning system for spider diagrams augmented with constants. The rules were largely extensions of those for spider diagrams without constants. However, some extensions were not straightforward, such as for the rule that allows the introduction of a contour and that which allows diagrams to be combined. In the latter case, this is due (in part) to the increased number of ways that two unitary diagrams can be inconsistent.

The proof of completeness is constructive, in that it provides an algorithm to convert a premise diagram into a conclusion diagram using the inference rules presented. Whilst we omitted details of the completeness proof due to space reasons, the proof strategy was a generalization of that for spider diagrams without constants. We note, though, that the actual details of the completeness proof are more complex when constants are involved.

In the future, we plan to integrate constant spiders into constraint diagrams. The work in this paper lays the foundations for developing a reasoning system for constraint diagrams augmented with constants. We also plan to investigate how to efficiently automate the search for readable proofs when using spider diagrams augmented with constants, following [5].

Acknowledgement This research is supported by EPSRC grant EP/E011160 for the *Visualization with Euler Diagrams* project.

References

[1] L. Choudhury and M. K. Chakraborty. On extending Venn diagrams by augmenting names of individuals. In *Proceedings of 3rd International Conference on the Theory and Application of Diagrams*, volume 2980 of *LNAI*, pages 142–146. Springer-Verlag, March 2004.

[2] R. Clark. Failure mode modular de-composition using spider diagrams. In *Proceedings of Euler Diagrams 2004*, volume 134 of *Electronic Notes in Theoretical Computer Science*, pages 19–31, 2005.

[3] R. DeChiara, U. Erra, and V. Scarano. VennFS: A Venn diagram file manager. In *Proceedings of Information Visualisation*, pages 120–126. IEEE Computer Society, 2003.

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

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

[7] P. Hayes, T. Eskridge, R. Saavedra, T. Reichherzer, M. Mehrotra, and D. Bobrovnikoff. Collaborative knowledge capture in ontologies. In *Proceedings of the 3rd International Conference on Knowledge Capture*, pages 99–106, 2005.

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

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

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

[11] H. Kestler, A. Muller, T. Gress, and M. Buchholz. Generalized Venn diagrams: A new method for visualizing complex genetic set relations. *Journal of Bioinformatics*, 21(8):1592–1595, 2005.

[12] 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.

[13] J. Lovdahl. *Towards a Visual Editing Environment for the Languages of the Semantic Web*. PhD thesis, Linkoping University, 2002.

[14] L. Niebrój. Defining health/illness: Societal and/or clinical medicine? *Journal of Physiology and Pharmacology*, 57(4):251–262, 2006.

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

[16] G. Stapleton and A. Delaney. Evaluating and generalizing constraint diagrams. *Journal of Visual Languages and Computing*, available online, 2008.

[17] G. Stapleton, J. Taylor, J. Howse, and S. Thompson. The expressiveness of spider diagrams augmented with constants. *Journal of Visual Languages and Computing*, available online, 2008.

[18] G. Stapleton, S. Thompson, J. Howse, and J. Taylor. The expressiveness of spider diagrams. *Journal of Logic and Computation*, 14(6):857–880, December 2004.

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

[20] J. Thièvre, M. Viaud, and A. Verroust-Blondet. Using euler diagrams in traditional library environments. In *Euler Diagrams 2004*, volume 134 of *ENTCS*, pages 189–202. ENTCS, 2005.

[21] Y. Zhao and J. Lövdahl. A reuse based method of developing the ontology for e-procurement. In *Proceedings of the Nordic Confernce on Web Services*, pages 101–112, 2003.