ARTICLE IN PRESS

Available online at www.sciencedirect.com

ScienceDirect

Journal of Visual Languages and Computing

▮ (▮▮▮▮) ▮▮▮–▮▮▮

Journal of Visual Languages & Computing

www.elsevier.com/locate/jvlc

The expressiveness of spider diagrams augmented with constants

Gem Stapleton[a,*,1,2,4], John Taylor[a,3], Simon Thompson[b,4], John Howse[a,2,3]

[a] *Visual Modelling Group, University of Brighton, Brighton, UK*
[b] *Computing Laboratory, University of Kent, Canterbury, UK*

Received 13 April 2007; received in revised form 17 December 2007; accepted 28 January 2008

Abstract

Spider diagrams are a visual language for expressing logical statements or constraints. Several sound and complete spider diagram systems have been developed and it has been shown that they are equivalent in expressive power to monadic first order logic with equality. However, these sound and complete spider diagram systems do not contain syntactic elements analogous to constants in first order predicate logic. We extend the spider diagram language to include *constant spiders* which represent specific individuals. Formal semantics are given for the extended diagram language. We prove that this extended system is equivalent in expressive power to the language of spider diagrams without constants and, hence, equivalent to monadic first order logic with equality.

Keywords: Diagrammatic logic; Visual formalism; Formal methods

*Corresponding author. Tel.: +44 1273 642410.
E-mail addresses: g.e.stapleton@brighton.ac.uk (G. Stapleton), john.taylor@brighton.ac.uk (J. Taylor), s.j.thompson@kent.ac.uk (S. Thompson), john.howse@brighton.ac.uk (J. Howse).
URL: http://www.brighton.ac.uk/cmis/research/vmg (G. Stapleton).
[1]This research was supported by a Leverhulme Trust Early Career Fellowship.
[2]This research was partially supported by UK EPSRC Grant EP/E011160.
[3]This research was partially supported by UK EPSRC Grant GR/R63516.
[4]This research was partially supported by UK EPSRC Grant GR/R63509.

1045-926X/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jvlc.2008.01.005

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.

Traditionally in mathematics and logic, diagrams have been excluded from playing a formal role and were considered only as a heuristic aid. Some people have held the view that diagrams *cannot be formalized*, so as to be permitted when reasoning formally. However, it has been shown that this view is incorrect: Shin devised a sound and complete diagrammatic logic [1] that is capable of making statements about certain relationships between sets. Her work is widely regarded as seminal, overturning the view that diagrams could not yield a formal reasoning system. Thus, diagrams are now being recognized as a valuable tool that can be exploited in a logical setting; see the overview paper [2] for an extensive discussion of the importance of diagrams in numerous reasoning contexts.

With such a large body of research existing for symbolic logics, there needs to be solid justification for developing diagrammatic logics. Whilst logicians and mathematicians are highly competent when using symbolic logics, including formulating rigorous arguments, they typically have many years training and experience of working in such a way. Unfortunately, symbolic logics are not generally accessible to a broad range of potential users due to the steep learning curve associated with the accurate and fluent use of 'mathematical' symbols.

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. Thus, symbolic logics do not provide a comprehensive solution to the problem of precisely specifying software in an accessible way. Perhaps this is a reason why there has not been any significant uptake of formal methods by the software engineering community in general. By contrast, there is extensive use of diagrams to model software, with the Unified Modelling Language (UML) [3] 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.

Constraint diagrams were introduced in [4] as a way to visualize object-oriented invariants in the context of the UML and were subsequently extended to depict operation contracts as well [5]. They have been used to develop high-level models independently of UML [6,7]. 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 Fig. 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 [8].

The language of spider diagrams [9,10] forms a fragment of the constraint diagram language. The only spiders present in spider diagrams represent the existence of elements (called *existential spiders*) and arrows are not permitted. The spider diagram d_3 in Fig. 2 expresses, by the disjointness of the curves *Fish* and *Lions*, that no element is both a fish

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

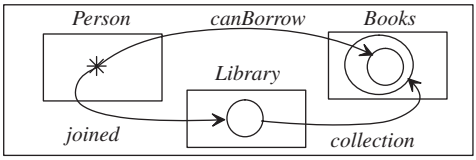

Fig. 1. A constraint diagram.

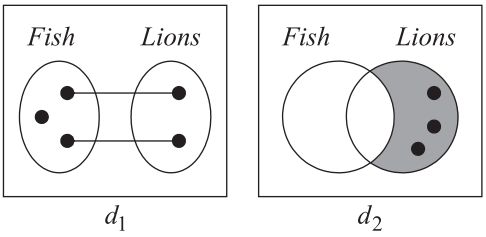

Fig. 2. Two spider diagrams.

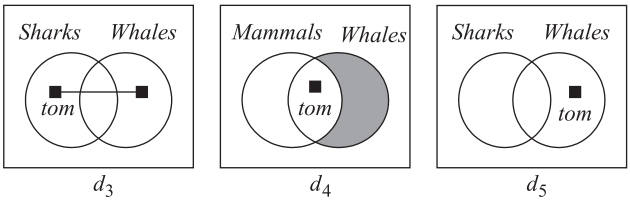

Fig. 3. Spider diagrams with constants.

and a lion and there are at least three elements, one of which is a fish, the other two are in the set *Fish* ∪ *Lions*. The spider diagram d_4 expresses that there are exactly three lions that are not fish. Shading is used to indicate an upper limit on the cardinality. It has been shown that the spider diagram language is equivalent in expressive power to monadic first order logic with equality [11].

It is not clear whether spider diagrams provide us with a mechanism for talking about particular, named, individuals. It would seem useful to introduce a syntactic device analogous to constant symbols in predicate logic. Indeed, from a usability perspective, it may be important to augment the language of spider diagrams with such syntactic devices. We introduce *constant spiders* (corresponding to *given spiders* in [12]) to provide users of the notation with explicit syntax with which to write constraints involving named individuals. At the syntactic level, we distinguish the two types of spiders by using round nodes for existential spiders and square nodes for constant spiders. Moreover, constant spiders will always be labelled and existential spiders will not be labelled.

In Fig. 3, the diagrams d_5, d_6 and d_7 all contain a constant spider labelled *tom*. The diagram d_5, for example, expresses that *tom* is either a shark or a whale, but not both. From the conjunction of d_5 and d_6 we can deduce that *tom* is a whale but not a shark, expressed by d_7 (that is, d_7 is a consequence of the conjunction of d_5 and d_6). By contrast, in Fig. 4, from d_8 and d_9, which contain existential spiders, we cannot deduce d_{10}.

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

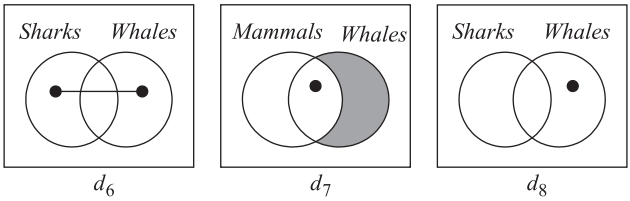

Fig. 4. Spider diagrams without constants.

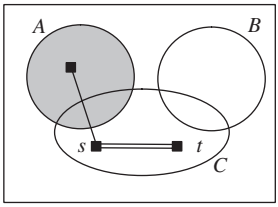

Fig. 5. A spider diagram containing a tie.

We also augment the language with *ties* in order to assert that two constants represent the same individual. A tie is a pair of parallel straight line segments that connect constant spiders. Any two nodes (called *feet*) can be joined by a tie provided that the two nodes are placed in the same minimal region called a *zone*. Two constant spiders, s_1 and s_2 say, joined by a tie represent the same individual if and only if s_1 and s_2 both represent an individual in a zone that contains a tie between them. For example, the diagram in Fig. 5 contains two constant spiders, s and t, that are joined by a tie. The diagram asserts that s represents an individual in the set $C - (A \cup B)$ if and only if s represents the same individual as t.

Previous spider diagram systems have not included explicit negation of diagrams. That is, for any spider diagram D it is not the case that $\neg D$ is a spider diagram. Our final extension to the syntax is to remove this restriction by incorporating the $\neg$ operator.

In Section 5, we show that the spider diagram language augmented with constants, ties and negation is expressively equivalent to the spider diagram language, thus proving that our extensions to the syntax do not increase expressiveness. A key idea of the proof is to turn each constant spider into a contour containing a single existential inhabitant. However, this key idea by itself is not sufficient; it merely points us towards a correct proof technique which has to be adapted to take into account a variety of issues. One issue of particular note is that we allow empty universes which is not typical. Further difficulties are discussed in Section 5 and we provide a thorough treatment of the translation required to eliminate constants, ties and negation from the language. Clearly introducing constant spiders, ties and negation does not decrease expressiveness and it follows that the language of spider diagrams with constants is equally as expressive as the language of spider diagrams and, hence, to monadic first order logic with equality; see [11].

We review related work and some application areas for spider diagrams in Section 2. In Section 3, we define the syntax of spider diagrams with constants and Section 4 formalizes the semantics.

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

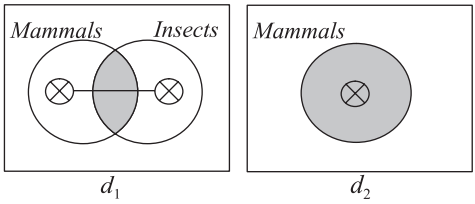

Fig. 6. Two Venn-II diagrams.

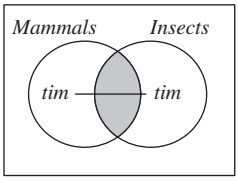

Fig. 7. An Euler/Venn diagram.

2. Related work and applications of spider diagrams

Several visual languages have emerged that extend Euler and Venn diagrams; for example, Venn-II introduced by Shin [1]. The diagram d_1 in Fig. 6 is a Venn-II diagram. In addition to what is expressed by the underlying Venn diagram (i.e. $Mammals \cap Insects = \emptyset$), d_1 also expresses, using an $\otimes$-*sequence*, the set $Mammals \cup Insects$ is not empty.

Venn-II diagrams can express whether a set is empty or not empty but cannot express arbitrary finite lower bounds on cardinality. So, the presence of more than one $\otimes$-sequence in a particular region provides no more information than a single $\otimes$-sequence in that region. Furthermore, if an $\otimes$-sequence is placed in the same region as shading in a diagram, then the diagram expresses contradictory information and is unsatisfiable. For example, the diagram d_2 in Fig. 6 asserts that $Mammals = \emptyset$ (by the use of shading) and $Mammals \neq \emptyset$ by the use of an $\otimes$-sequence and, therefore, has no models. Shin shows that Venn-II is equivalent in expressive power to monadic first order logic (in which all predicate symbols are one place) and she calls this language $\mathscr{L}_0$ [1]. The language $\mathscr{L}_0$ is a pure monadic language that does not include equality, constants or function symbols. Shin also defined sound and complete reasoning rules for Venn-II.

In [13], Swoboda and Allwein introduce their Euler/Venn language, based on Euler diagrams. Euler/Venn diagrams do not contain $\otimes$-sequences but instead use *constant sequences* to talk about particular individuals rather than simply denoting the non-emptiness of a set. Another difference is that Euler/Venn diagrams have underlying Euler diagrams whereas Venn-II diagrams are more restrictive, allowing only Venn diagrams as the underlying diagrams. The diagram in Fig. 7 is an Euler/Venn diagram and expresses that no element is both a mammal and an insect and that there is something called *tim* that is either a mammal or an insect. The semantics of constant sequences (used in Euler/Venn diagrams) are different from our interpretation of constant spiders: both represent particular individuals but, within a diagram, constant sequences with distinct labels do not necessarily denote distinct individuals whereas

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

constant spiders with distinct labels do denote distinct individuals, unless they are joined by a tie.

Swoboda and Allwein give an algorithm that determines whether particular monadic first order formulas are 'observable' from a given Euler/Venn 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). In [14], sound reasoning rules for Euler/Venn diagrams are given.

In [11,15] we proved that the spider diagram language without constants is equivalent in expressive power to monadic first order logic with equality ($\mathscr{MFOL}_=$). The language $\mathscr{MFOL}_=$ extends $\mathscr{L}_0$ by adding equality. Within $\mathscr{L}_0$ it is not possible to express that a particular property, P, holds for a unique element, whereas this is straightforward in $\mathscr{MFOL}_=$:

$$\exists x(P(x) \wedge \forall y(P(y) \Rightarrow x = y)).$$

Thus spider diagrams properly increase expressiveness over Venn-II diagrams.

Sound and complete reasoning rules for various spider diagram systems without constants have been given [9,10] (differing, for example, by way of being based on either Euler or Venn diagrams). A sound, but not complete, system of spider diagrams that includes constant spiders, but not existential spiders, can be found in [12]. The reasoning rules presented in [12] are largely similar to those in [9,10]. However, the level of rigour displayed in the formalization of the most recent spider diagram system, namely that in [9], is much higher than that in [12]; a key difference is the use of a very precise abstract syntax in [9] as opposed to the concrete syntax specified in [12]. Consequently, there is a need to put spider diagrams with constants on an 'equal footing' to those without constants, which we do in this paper.

In [16], we first considered the impact of augmenting spider diagrams with constants. The approach taken utilized a non-standard definition of the semantics of constants: when the universal set was non-empty we did not force constants to denote. This leads to some counter-intuitive aspects that were not apparent until one considers reasoning with the notation. In particular, from a diagram containing a single existential spider only (i.e. asserting the non-emptiness of the universe), one could not (semantically) deduce that the individual *tim* was in the universe, even though we may have some syntactical device for representing *tim*. Drawing an analogy with representing sets, this would be similar to having access to a contour (closed curve) label, such as 'mammals', and not being able to deduce from the knowledge that the universe is non-empty that there is an element which is either a mammal or not a mammal. Thus, in this paper, we improve the semantics for constants given in [16] so that these non-intuitive situations do not arise.

Moreover, [16] allowed an overloading of label use: any given label could be used to label a contour in one diagram and a constant spider in another. In this paper, we make the distinction between sets and individuals more explicit at the syntax level and use one set of labels specifically for contours and another (disjoint) set of labels for constants. This impacts our definition of spider diagrams with constants given below. These changes to the syntax and semantics have a significant impact on the proof that constants do not lead to an increase in expressive power. Indeed, the details of the proof become much more complex. A number of additional results are also provided in this paper which do not appear in [16]. First, [16] does not include ties; not only do we provide a formalization of the notation involving ties, but we also prove that they can be removed from the notation

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

without decreasing expressiveness. Secondly, we now prove that removing the negation operator also does not lead to a decrease in expressiveness, a result some may find surprising since diagrammatic languages are often thought not rich enough to express negated statements. Finally, we also provide a satisfiability result, showing how to construct a model for any so-called unitary spider diagram with constants.

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 [17]. They have also been used (but not explicitly) for displaying the results of database queries [18], representing non-hierarchical computer file systems [19], in a visual semantic web editing environment [20,21] and for viewing clusters which contain concepts from multiple ontologies [22]. All of these application areas (except the first) use constants to represent specific objects, thus highlighting the importance of augmenting spider diagrams with constants.

3. Syntax

In this section, we define what constitutes a spider diagram, using an abstract syntax. There are good, well documented reasons for using this type of approach, rather than defining at the concrete (drawn) diagram level; see, for example, [23,24].

The *contour labels* (that is, the closed curves' labels) used in our diagrams are chosen from a countably infinite set, $\mathscr{CL}$. Informally, a *zone* is a region of the plane that can be described by the set of labels of the contours that include it. However, in different diagrams, zones can be included by contours with the same labels but differ in the labels which exclude them. We will define a *zone* to be a pair of finite, disjoint sets, (in, ex). The set *in* contains the labels of the contours that include (in, ex) whereas *ex* is the set of labels of the contours that do not include (in, ex). So, in a diagram, *in* and *ex* form a partition of the contour label set. A *region* is a non-empty set of zones. We define $\mathscr{Z}$ and $\mathscr{R} = \mathbb{P}\mathscr{Z} - \{\emptyset\}$ to be the sets of all zones and regions, respectively. As an example, the diagram in Fig. 8, contains two zones, $(\{A\}, \emptyset)$ and $(\emptyset, \{A\})$, and therefore contains three regions.

To describe the existential spiders in a drawn diagram, it is sufficient to say how many existential spiders there are in each region. In Fig. 8, for example, there is one existential spider in the region $\{(\{A\}, \emptyset)\}$ and another in the region $\{(\{A\}, \emptyset), (\emptyset, \{A\})\}$. The nodes of the spiders are called *feet*; there are two one-footed spiders and a two-footed spider in d_1. We will use a bag of regions, called *existential spider descriptors*, to formalize the notion of an existential spider. Alternatively, we could specify any finite set to be a collection of existential spiders and map each of these spiders to a region in the diagram (called the 'habitat mapping', with the region in which the spider is placed called its habitat). However, with this alternative choice, for any given drawn diagram containing existential

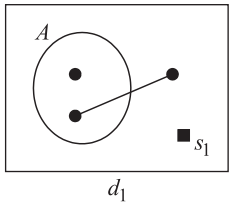

Fig. 8. The syntax of spider diagrams.

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

spiders there are many choices for the representation of the drawn spiders at the abstract level.

In any diagram we use only a finite set of constant spiders. We will assume that all the constant spider labels come from a finite set $\mathscr{CS}$. An alternative choice would be to have a countably infinite set of constant spider labels. However, the approach we take to prove that augmenting the spider diagram language with constants does not increase expressiveness would need to be adjusted if $\mathscr{CS}$ is not finite and we discuss this at the end of Section 5.

Formally, a diagram will contain a finite set of constant spider labels together with a habitat function, mapping each constant spider label to a region in the diagram. The definition of an abstract spider diagram with constants extends that given in [15] for spider diagrams (without constants). We assume that the sets $\mathscr{CS}$, $\mathscr{CL}$, $\mathscr{Z}$ and $\mathscr{R}$ are pairwise disjoint.

Now we are in a position to specify formally spider diagrams with constants. Example 3.1 will more fully illustrate the concepts. We start by defining so-called *unitary* diagrams and then extend the definition to allow such diagrams to be joined using logical connectives.

Definition 3.1. An *abstract unitary spider diagram with constants*, d (with contour labels in $\mathscr{CL}$ and constant spider labels in $\mathscr{CS}$) is a tuple $\langle L, Z, Z^*, ESD, CS, \theta, \omega\rangle$ whose components are defined as follows.

(1) $L = L(d) \subset \mathscr{CL}$ 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 contour 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 $Z(d)$.

   We define $R(d) = \mathbb{P}Z(d) - \{\emptyset\}$ to be the set of regions in d.
(3) $Z^* = Z^*(d) \subseteq Z$ is a set of *shaded zones* and we define $R^*(d) = \mathbb{P}Z^*(d) - \{\emptyset\}$ to be the set of shaded regions in d.
(4) $ESD = ESD(d) \subset \mathbb{Z}^+ \times R(d)$ is a finite set of *existential spider descriptors* such that

$$\forall (n_1, r_1), (n_2, r_2) \in ESD\ (r_1 = r_2 \Rightarrow n_1 = n_2).$$

   If $(n, r) \in ESD$ then there are n *existential spiders* with *habitat* r.
(5) $CS = CS(d) \subseteq \mathscr{CS}$ is a finite set of constant spider labels.
(6) $\theta = \theta_d : CS \to R(d)$ is a function which maps each constant spider label to a region in d. If $\theta_d(s_i) = r$ then s_i has *habitat* r in d.
(7) $\omega = \omega_d : CS \times CS \to \mathbb{P}Z$ is a function which returns the *web* of each pair of constant spiders such that

$$\forall s_i, s_j, s_k \in CS\ \omega(s_i, s_j) \subseteq \theta(s_i) \cap \theta(s_j) \wedge \omega(s_i, s_i) = \theta(s_i)$$
$$\wedge\ \omega(s_i, s_j) = \omega(s_j, s_i) \wedge (\forall z \in Z(d)\ (z \in \omega(s_i, s_j) \cap \omega(s_j, s_k) \Rightarrow z \in \omega(s_i, s_k))).$$

   The web of a pair of constant spiders is the set of zones that contain a *tie* between those two spiders.

Let $d = \langle L, Z, Z^*, ESD, CS, \theta, \omega\rangle$ be a unitary spider diagram with constants. The tuple $\langle L, Z, Z^*, ESD\rangle$ is a *unitary spider diagram without constants*.

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

Some remarks about the above definition are in order. Every contour in a diagram contains at least one zone and this is captured by condition 2(i). In any diagram, the zone inside the boundary rectangle but outside all the contours is present and this is captured by condition 2(ii). Being joined by a tie in a zone is interpreted transitively. In fact, ties give rise to an equivalence relation on the feet in each zone. In the abstract syntax, if spiders s_i and s_j are joined by a tie in zone z and s_j and s_k are also joined by a tie in z then so too are s_i and s_k, giving the transitive property. Moreover, s_i is deemed to be joined by a tie to itself in each zone of its habitat, giving the reflexive property. Finally, for symmetry, s_i is joined to s_j in zone z if and only if s_j is joined to s_i in zone z. Therefore, in a zone z, taking the constant spider feet in z as a set of vertices and the ties in that zone as a set of edges, we have a graph whose components are complete graphs with loops at each vertex. However, in drawn diagrams we will only draw a spanning forest in each zone so as to avoid 'visually cluttered' diagrams.

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 types of tie are permitted.

Example 3.1. The diagram d_1 in Fig. 9 has the following formal description:

(1) Contour label set $L(d_1) = \{L_1, L_2\}$.
(2) Zone set

$$Z(d_1) = \{(\emptyset, \{L_1, L_2\}), (\{L_1\}, \{L_2\}), (\{L_2\}, \{L_1\}), (\{L_1, L_2\}, \emptyset)\}.$$

(3) Shaded zone set $Z^*(d_1) = \{(\{L_2\}, \{L_1\})\}$.
(4) Existential spider descriptors set

$$ESD(d_1) = \{(1, \{(\{L_2\}, \{L_1\})\}), (1, \{(\{L_1\}, \{L_2\}), (\{L_2\}, \{L_1\})\})\}.$$

(5) Constant spider label set $CS(d_1) = \{s_1, s_2\}$.
(6) The function $\theta_{d_1}: \{s_1, s_2\} \to R(d_1)$ where $\theta_{d_1}(s_1) = \{(\{L_1\}, \{L_2\})\}$ and $\theta_{d_1}(s_2) = \{(\{L_1, L_2\}, \emptyset)\}$.
(7) The function $\omega_{d_1}: CS(d_1) \times CS(d_1) \to \mathbb{P}Z(d_1)$ where $\omega_{d_1}(s_1, s_1) = \theta_{d_1}(s_1)$, $\omega_{d_1}(s_2, s_2) = \theta_{d_1}(s_2)$ and $\omega_{d_1}(s_1, s_2) = \omega_{d_1}(s_2, s_1) = \emptyset$.

In order to be able to refer to the set of existential spiders in a diagram, d, we define

$$ES(d) = \{e_i(r) : \exists (n, r) \in ESD(d) \wedge 1 \leqslant i \leqslant n\}$$

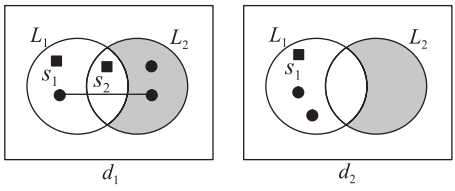

Fig. 9. Two spider diagrams with constants.

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

to be the set of *existential spiders*. The subscript 'i' can be thought of as labelling the existential spiders in a region. We also define $S(d) = ES(d) \cup CS(d)$ to be the set of *spiders* in d. We assume that the sets $ES(d)$ and $\mathscr{CS} \cup \mathscr{CL} \cup \mathscr{L} \cup \mathscr{R}$ are disjoint. We define a function

$$\eta: ES(d) \to R(d)$$

by $\eta(e_i(r)) = r$ which returns the *habitat* of each existential spider. Spiders represent the existence of elements and regions represent sets—thus we need to know how many elements we have represented in each region. Note here that, in a unitary diagram, a constant spider and an existential spider represent the existence of distinct elements. For example, in Fig. 9, the diagram d_2 asserts that the set represented by the zone $(\{L_1\}, \{L_2\})$ contains at least three elements, including the individual represented by s_1. The set of existential spiders contained by region r in d is denoted by $ES(r, d)$. More formally,

$$ES(r, d) = \{e \in ES(d) : \eta(e) \subseteq r\}.$$

Similarly, the set of constant spiders contained by region r in d is

$$CS(r, d) = \{s \in CS(d) : \theta_d(s) \subseteq r\}$$

and we also define $S(r, d) = ES(r, d) \cup CS(r, d)$. So, any spider in d whose habitat is a subset of r is in the set $S(r, d)$. The set of existential spiders *touching* r in d is denoted by $ET(r, d)$:

$$ET(r, d) = \{s \in ES(d) : \eta(s) \cap r \neq \emptyset\}.$$

The sets of constant spiders touching a region, $CT(r, d)$, and the set of spiders touching a region, $T(r, d)$, are defined similarly. In d_1, Fig. 9,

$$|S(\{(\{L_2\}, \{L_1\})\}, d_1)| = 1$$

and

$$|T(\{(\{L_2\}, \{L_1\})\}, d_1)| = 2.$$

In d_2,

$$|S(\{(\{L_1\}, \{L_2\})\}, d_2)| = |T(\{(\{L_1\}, \{L_2\})\}, d_2)| = 3.$$

Unitary diagrams form the building blocks of *compound diagrams*.

Definition 3.2. An *abstract spider diagram with constants* is defined as follows.

(i) Any unitary diagram with constants is a spider diagram with constants.
(ii) If D_1 and D_2 are spider diagrams with constants then $\neg D_1$, $(D_1 \vee D_2)$ and $(D_1 \wedge D_2)$ are a spider diagrams with constants.

We adopt the usual convention of omitting brackets where no ambiguity arises. Another convention will be to denote unitary diagrams by d and arbitrary diagrams by D. Definition 3.2 adapts to spider diagrams without constants in the obvious way.

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

4. Semantics

We now sketch, informally, the semantics of unitary spider diagrams. Regions in spider diagrams with constants represent sets. Missing zones (i.e. zones in the set $\{(a,b) \in \mathscr{Z} : a \cup b = L(d)\} - Z(d)$) represent the empty set. Existential spiders assert the existence of elements and distinct existential spiders assert the existence of distinct elements. Therefore, we can express lower and, using shading, upper bounds on the cardinalities of the sets we are representing. For simplicity, suppose the diagram d does not contain any ties. If region r is inhabited by n spiders in 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.

Each constant spider asserts that the individual represented by its label is in the set represented by its habitat. Moreover, the individuals represented by constant spiders are distinct from those represented by existential spiders. Therefore, if a region contains an existential spider and a constant spider, s, we can deduce that there are at least two elements in that region, including that represented by s. Within a unitary diagram, no two constant spiders represent the same individual unless they are joined by a tie. Constant spiders joined by ties must represent the same individual if they both represent individuals in the set represented by some particular zone in their web, otherwise they must represent distinct individuals. So, the presence of a tie between two constant spiders has the effect of potentially reducing the upper and lower cardinality constraints placed on the set represented by the union of their habitats.

To formalize the semantics of spider diagrams with constants we shall map the constant spider labels in $\mathscr{CS}$, the contour labels in $\mathscr{CL}$, zones in $\mathscr{Z}$ and regions in $\mathscr{R}$ to subsets of some universal set U. We wish constant spider labels to act like constants in first order predicate logic, so they will be interpreted by single element subsets of the universal set, unless the universal set is the empty set. We could, equivalently, choose to map constant spiders to elements of the universal set. However, the *semantics predicate* (defined below) is more elegant when we map constant spiders to sets, as are the details in some of the proofs below. We could also choose to force models with constant spiders to have non-empty universal sets. However, having only existential spiders in the language does not force the universal set to be non-empty. We note that any unitary diagram containing spiders has only non-empty models. In either spider diagram language (with or without constants) we can express that there are no elements by shading all the zones in a unitary diagram that does not contain any spiders. The motivation for this non-standard choice (allowing an empty universe) arises from an intended application domain of constraint diagrams: modelling object-oriented systems. The domain will consist of objects in the system and in some instances there will be no objects (for example, in an initial state before any objects have been created). Logic with potentially empty structures is explored in [25].

Our formalization of the semantics extends that given for spider diagrams without constants in [15].

Definition 4.1. An *interpretation of constant spider labels*, *contour labels*, *zones and regions*, or simply an *interpretation with constants*, is a pair (U, Ψ) where U is a set (the *universal set*) and $\Psi: \mathscr{CL} \cup \mathscr{Z} \cup \mathscr{R} \cup \mathscr{CS} \to \mathbb{P}U$ is a function mapping constant spider labels, contour

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

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 either the universal set is the empty set or the constant spiders map to singleton subsets of U. More formally

$$U = \emptyset \vee \forall s_i \in \mathscr{CS}\ |\Psi(s_i)| = 1.$$

We will write $\Psi: \mathscr{R} \cup \mathscr{CS} \to \mathbb{P}U$ when strictly speaking we mean $\Psi: \mathscr{CL} \cup \mathscr{Z} \cup \mathscr{R} \cup \mathscr{CS} \to \mathbb{P}U$.

We introduce a *semantics predicate* which determines whether an interpretation agrees with the meaning of any given diagram with constants.

Definition 4.2. Let D be a diagram with constants and let $m = (U, \Psi)$ be an interpretation with constants. We define the *semantics predicate* of D, denoted $P_D(m)$. If D is a unitary diagram then $P_D(m)$ is the conjunction of the following conditions.

(1) *Plane tiling condition*: The union of the sets represented by the zones in D is the universal set:

$$\bigcup_{z \in Z(D)} \Psi(z) = U.$$

(2) There exists an extension of $\Psi: \mathscr{R} \cup \mathscr{CS} \to \mathbb{P}U$ to $\Psi: \mathscr{R} \cup \mathscr{CS} \cup ES(D) \to \mathbb{P}U$ such that the following conditions are satisfied.
   (a) *Spiders condition*: Each spider represents the existence of an element (strictly, a single element set) in the set represented by its habitat and existential spiders do not represent the same elements as any constant spiders:

$$\forall s \in ES(D)\ (|\Psi(s)| = 1 \wedge \Psi(s) \subseteq \Psi(\eta(s)))$$

and

$$\forall s \in CS(D)\ (|\Psi(s)| = 1 \wedge \Psi(s) \subseteq \Psi(\theta_D(s)))$$

and

$$\forall e \in ES(D)\ \forall s \in CS(D) \Psi(e) \neq \Psi(s).$$

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

(b) *Existential spiders condition*: No two existential spiders represent the existence of the same element:

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

That is, the function Ψ is injective when the domain is restricted to $ES(D)$.

(c) *Constant spiders condition*: Two constant spiders represent the same individual if and only if they both represent an individual in the set denoted by some zone in their web:

$$\forall s_i, s_j \in CS(D)\ (\Psi(s_i) = \Psi(s_j) \ \Leftrightarrow\ \exists z \in \omega_D(s_i, s_j)\Psi(s_i) \cup \Psi(s_j) \subseteq \Psi(z)).$$

(d) *Shading condition*: Each shaded zone, z, represents a subset of the set of elements represented by the spiders touching z:

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

If $\Psi: \mathscr{R} \cup \mathscr{C}\mathscr{S} \cup ES(D) \to \mathbb{P}U$ ensures $P_D(m)$ is true then Ψ is a *valid extension to existential spiders* for D. If $D = \neg D_1$ then $P_D(m) = \neg P_{D_1}(m)$. If $D = (D_1 \vee D_2)$ then $P_D(m) = (P_{D_1}(m) \vee P_{D_2}(m))$. If $D = (D_1 \wedge D_2)$ 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.

For example, the interpretation $m = (\{1, 2, 3, 4\}, \Psi)$ partially defined by $\Psi(s_1) = \{1\}$, $\Psi(s_2) = \{2\}$, $\Psi(L_1) = \{1, 2\}$ and $\Psi(L_2) = \{2, 3, 4\}$ is a model for d_1 in Fig. 9 but not for d_2.

From the definition of the semantics predicate, it follows that a unitary diagram, d, has an empty model if and only if d does not contain any spiders. It is easy to see that for any unitary spider diagram without constants, the constant spiders condition is always true. We make the following definitions for the language of spider diagrams without constants.

Definition 4.3. Let $m = (U, \Psi)$ be an interpretation with constants. We restrict the domain of Ψ to $\mathscr{C}\mathscr{L} \cup \mathscr{L} \cup \mathscr{R}$ to give the pair $(U, \Psi|_{\mathscr{C}\mathscr{L} \cup \mathscr{L} \cup \mathscr{R}})$ which we call an *interpretation*.

Making the obvious change to the semantics predicate given for spider diagrams with constants, we define the *semantics predicate* for spider diagrams without constants. The definitions of *satisfies* and *model* adapt similarly. We note here that the semantics predicate we give for spider diagrams without constants is different from that given in [11], which uses a collection of inequalities to capture the notion of a model,[5] however, they are equivalent [26]. That is, the two semantics predicates identify the same interpretations as models for any given diagram without constants.

Theorem 4.1. *Every unitary diagram d is satisfiable.*

Proof (*Sketch*). The proof strategy is to construct a model for d. We start by noting that, in any model for unitary diagram d, the set represented by the region containing all of the zones is the universal set. Moreover, there must be sufficiently many elements in the universal set for the spiders in d, taking any ties into account. To start the construction of

[5]The semantics predicate in [11] states that for a unitary diagram d, the number of existential spiders in any given region, r, in d is at least $|\Psi(r)|$, and, if r is shaded, $|\Psi(r)|$ is at most the number of existential spiders touching r.

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

our model, we specify the universal set as follows. First, define a function $f: S(d) \to Z(d)$ so that for each spider s, $f(s)$ is in the habitat of s, essentially selecting a foot of each spider. Recall that ω_d identifies which spider feet are joined by ties. For each constant spider, s_i, we define

$$[s_i] = \{s_j \in CS(d) : f(s_j) = f(s_i) \wedge f(s_i) \subseteq \omega_d(s_i, s_j)\}.$$

It is easy to verify that these sets $[s_i]$ give rise to an equivalence relation and, hence, form a partition of $CS(d)$. The universal set is then taken to be

$$U = ES(d) \cup \{[s_i] : s_i \in CS(d)\}.$$

Next, we define Ψ. Each contour label, L, in d maps to the set

$$\Psi(L) = \{e \in ES(d) : f(e) = (a, b) \wedge L \in a\} \cup \{[s_i] : s_i \in CS(d) \wedge f(s_i) = (a, b) \wedge L \in a\}$$

and each constant spider, s_k, in d, maps to the set

$$\Psi(s_k) = \{[s_k]\}.$$

The constant spiders (labels) that are not in $CS(d)$ map to any single element subset of U, provided U is not empty. The contour labels that are not in $L(d)$ map to any subset of U. It is relatively straightforward to show that (U, Ψ) is a model for d, noting that for each zone, z, in d, $\Psi(z) = \{e \in ES(d) : f(e) = z\} \cup \{[s_i] : s_i \in CS(d) \wedge f(s_i) = z\}$. □

We note here that the equivalence relation on $CS(d)$ induced by f in the above proof is related to ω_d but not equivalent to it. For each choice of f, there is one such equivalence relation which essentially identifies when the spider feet selected by f are joined by a tie. When each spider in d has a single foot only, the equivalence relation on $CS(d)$ induced by f and ω_d are capturing the same information; in such a case, f is unique. Much of the work in the remainder of this paper considers diagrams where each spider has a single foot only and the equivalence classes $[s_i]$ are utilized.

5. Expressiveness

In order to show that augmenting spider diagrams with constants does not increase expressiveness, we will specify a translation from spider diagrams with constants to spider diagrams without constants ensuring that an *expressively equivalent* relation holds. Informally, two languages are equivalent in expressive power when they are capable of axiomatizing the same classes of interpretations (sometimes called structures), up to some notion of equivalence between interpretations; this will be more fully explored shortly.

The essence of our translation, for unitary diagrams, is to replace each constant spider by a contour containing a single existential spider, shading and nothing else. The associated contour label is determined by the label of the constant spider; a function, $\mathscr{L}$, will be defined which maps elements of $\mathscr{CS}$ to contour labels in order to enable consistent contour label selection across different unitary diagrams. Intuitively, a contour, L, with shading and an existential spider allows the identification of a particular individual, since in any model, (U, Ψ), there is only one element in $\Psi(L)$. There are some difficulties to be overcome, however; the aforementioned intuition points us towards a key technique used to eliminate constant spiders but is not adequate to cope with a variety of complicating issues.

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

First, we observe that a unitary diagram containing, say, two constant spiders with many feet, some of which are joined by ties, contains disjunctive information about situations when the two constant spiders denote the same individual. Incorporating this type of uncertainty into our translation of unitary diagrams makes the details more complicated. Our translation, therefore, will focus only on diagrams where the spiders have single feet (every diagram can be reduced to a semantically equivalent diagram in this form).

Secondly, in any given unitary diagram, d, it need not be the case that all of the constant spider labels are used (i.e $CS(d) \neq \mathscr{CS}$). However, in any non-empty model for d, all of the constant spiders labels in $\mathscr{CS}$ represent single element sets. Thus our translation must ensure that the contour labels, arising from the constant spider labels in $\mathscr{CS}$ under the function $\mathscr{L}$ just described, *all* represent single element sets, not just those arising from the constant spiders in d.

Third, it is not the case that, having translated unitary diagrams, we can extend the translation to compound diagrams inductively. This is because, for example, the negation of the translation of unitary diagram d is not expressively equivalent to $\neg d$. Intuitively, the negation of a diagram containing only a contour label, l, which is inhabited by a single spider and is shaded (in particular, those arising from a constant spider, c say) allows l to contain any number of elements other than exactly one but c always represents an individual unless $U = \emptyset$ regardless of whether a negated statement is made. To simplify the presentation of our results, we will first translate the fragment of spider diagrams with constants where the operator $\neg$ is not permitted.

In order to define when two diagrams are expressively equivalent we will now return to the notion of identifying when two interpretations are equivalent. In first order predicate logic, a structure (see, for example [25]) corresponds to our notion of an interpretation. A structure for a first order language with constant symbols consists of a universal set (or *domain*), U, together with an ordered list of sets, each of which corresponds to the interpretation of either a function symbol or a predicate symbol in the language. So, a structure can be written as

$$\langle U, f_1, f_2, \ldots, f_m, \ldots, R_1, R_2, \ldots, R_n, \ldots \rangle,$$

where each f_i is a function with domain $U^{arity(f_i)}$ and codomain U and each R_i is a subset of $U^{arity(R_i)}$.

Two structures are equal when they have the same universal set and the same ordered list of sets (i.e. the functions and relations). This notion of equality, therefore, is independent of the actual symbols being interpreted. Two first order predicate logic languages are equivalent in expressive power precisely when they can axiomatize the same sets of structures under this notion of equality. We generalize this notion to the spider diagrams case.

We will assume, without loss of generality, throughout this section that $\mathscr{CL} = \{L_1, L_2, \ldots, L_n, \ldots\}$ and $\mathscr{CS} = \{s_1, s_2, \ldots, s_m\}$. Given an interpretation with constants, $I = (U, \Psi)$, we can write I in a similar manner to structures (i.e. as an ordered list), provided we consider Ψ as its image, rather than as a function:

$$I = \langle U, \Psi(s_1), \Psi(s_2), \ldots, \Psi(s_m), \Psi(L_1), \ldots, \Psi(L_n), \ldots \rangle.$$

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

Likewise, we can write an interpretation (without constants), $J = (V, \Phi)$ as

$$J = \langle V, \Phi(L_1), \Phi(L_2), \ldots, \Phi(L_m), \Phi(L_{m+1}), \ldots, \Phi(L_{m+n}), \ldots \rangle.$$

As just stated, the actual labels interpreted are of no significance when considering equality of structures. Generalizing this idea to interpretations, I is equivalent to J precisely when $U = V$, $\Psi(s_i) = \Phi(L_i)$ for all constant spiders s_i and $\Psi(L_i) = \Phi(L_{m+i})$ for all L_i, illustrated below:

$$\begin{array}{ccccccccccccc} I & = & \langle & U, & \Psi(s_1), & \Psi(s_2), & \ldots, & \Psi(s_m), & \Psi(L_1), & \ldots, & \Psi(L_n), & \ldots & \rangle \\ & & & \| & \| & \| & \ldots & \| & \| & \ldots & \| & \ldots & . \\ J & = & \langle & V, & \Phi(L_1), & \Phi(L_2), & \ldots, & \Phi(L_m), & \Phi(L_{m+1}), & \ldots, & \Phi(L_{m+n}), & \ldots & \rangle \end{array}$$

This is a mechanism we use to show that augmenting spider diagrams with constants and ties does not lead to an increase in expressive power.

Example 5.1. Suppose that $\mathscr{CS} = \{s_1\}$ (that is, $m = 1$) and consider the diagram d_1 in Fig. 10, which includes a constant spider. Our aim is to find a spider diagram without constants expressively equivalent to d_1. To construct such a diagram, firstly we replace each contour label L_i by L_{i+1}. This 'frees' the contour label L_1. We can use this free contour label to identify a specific individual (constant symbols represent specific individuals). We replace the constant spider, s_1, by a contour with label L_1, that is entirely shaded inside and that contains a single existential spider. The resulting spider diagram without constants is d_2.

In general, we have $|\mathscr{CS}| = m$, so L_i will be freed for each $1 \leqslant i \leqslant m$.

Definition 5.1. Define a bijection $\mathscr{L}: \mathscr{CS} \cup \mathscr{CL} \to \mathscr{CL}$ by

$$\mathscr{L}(x_i) = \begin{cases} L_{i+m} & \text{if } x_i \in \mathscr{CL}, \\ L_i & \text{if } x_i \in \mathscr{CS}. \end{cases}$$

The codomain of an interpretation is a power set and we allow the power set of any set (including the empty set) to be a codomain. We define $\mathscr{U}$ to be the class of all sets.

Definition 5.2. Define $\mathscr{INT}_{\mathscr{CS}}$ to be the class of all interpretations with constants, that is

$$\mathscr{INT}_{\mathscr{CS}} = \{(U, \Psi) : U \in \mathscr{U} \wedge \Psi: \mathscr{CS} \cup \mathscr{CL} \cup \mathscr{Z} \cup \mathscr{R} \to \mathbb{P}U\},$$

where (U, Ψ) is an interpretation with constants. Define also $\mathscr{INT}_{\mathscr{ES}}$ to be the class of all interpretations ($\mathscr{ES}$ for existential spiders), that is

$$\mathscr{INT}_{\mathscr{ES}} = \{(U, \Psi|_{\mathscr{CL} \cup \mathscr{Z} \cup \mathscr{R}}) : (U, \Psi) \in \mathscr{INT}_{\mathscr{CS}}\}.$$

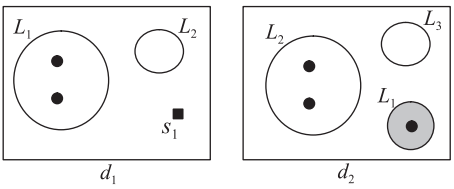

Fig. 10. A spider diagram with constants and an expressively equivalent spider diagram without constants.

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

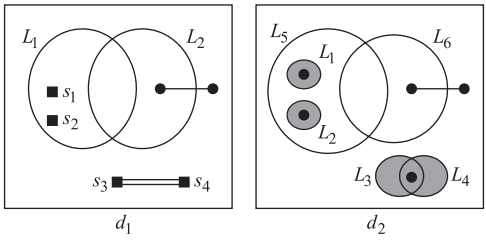

Fig. 11. Expressively equivalent diagrams.

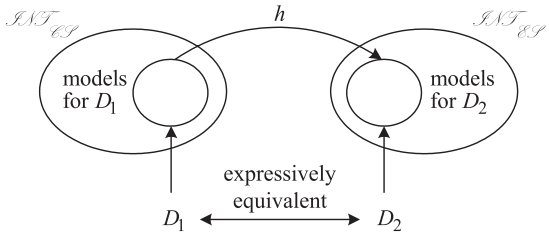

Fig. 12. Illustrating the relationship between models for expressively equivalent diagrams.

Using the function $\mathscr{L}$ we will now define a mapping, h, from interpretations with constants to interpretations which captures the notion of equivalence described above.

Definition 5.3. Define $h: \mathscr{INT}_{\mathscr{CS}} \to \mathscr{INT}_{\mathscr{ES}}$ by $h(U, \Psi) = (U, \Phi)$ where $\Phi: \mathscr{CL} \cup \mathscr{Z} \cup \mathscr{R} \to \mathbb{P}U$ is defined by $\Phi(L_i) = \Psi(\mathscr{L}^{-1}(L_i))$.

If, under Ψ, we consider the images of the elements in $\mathscr{CS} \cup \mathscr{CL} \cup \mathscr{Z} \cup \mathscr{R}$ as an ordered list, then applying h to (U, Ψ) will preserve this list.

Lemma 5.1. *The function h is injective.*

Example 5.2. For this example, assume that $\mathscr{CS} = \{s_1, s_2, s_3, s_4\}$. In Fig. 11 diagrams d_1 and d_2 are expressively equivalent. The function h provides a bijective correspondence between their models.

Whether two diagrams are expressively equivalent will be determined by the function h just defined. There are many other choices we could have made for h, each choice giving rise to an expressively equivalent relation.

Definition 5.4. Let D_1 be a spider diagram with constants and let D_2 be a spider diagram without constants. The diagrams D_1 and D_2 are *expressively equivalent* if and only if h provides a bijective correspondence between their models.

A model level relationship between expressively equivalent diagrams is shown in Fig. 12.

In the examples we have given so far to illustrate the expressively equivalent relation, we assumed that the constant spider label set $\mathscr{CS}$ was the same as the constant spider label set in the example diagrams. We now give a further illustration, but where $\mathscr{CS}$ contains more labels than the example diagram.

Example 5.3. For this example, assume that $\mathscr{CS} = \{s_1, s_2, s_3\}$. In Fig. 13 the diagram d_1 contains just s_1 and s_2. However, in any model for d_1, the constant spider s_3 represents a

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

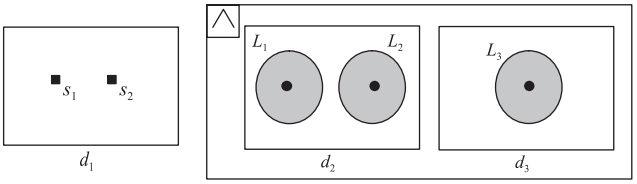

Fig. 13. Expressively equivalent diagrams.

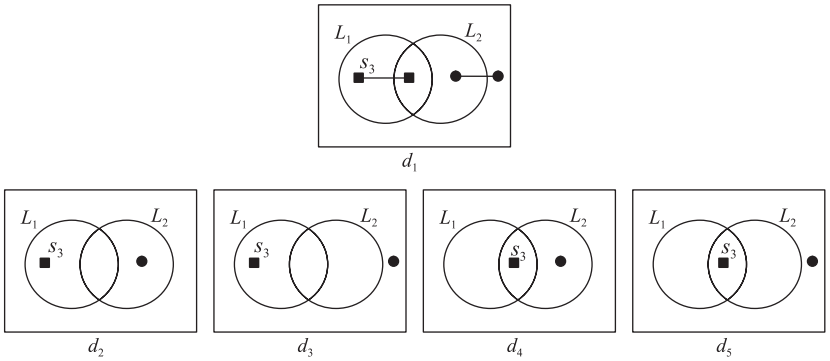

Fig. 14. Semantic equivalence.

specific individual. Thus, to find a diagram expressively equivalent to d_1, we must ensure that L_3 represents a single element set, asserted by d_3. The diagram $d_2 \wedge d_3$ is expressively equivalent to d_1.

In order to show that augmenting spider diagrams with constants does not increase expressiveness, we must find, for each spider diagram with constants, an expressively equivalent spider diagram without constants. To make this task more straightforward we appeal to *α-diagrams*. A spider diagram D (with or without constants) is called an *α-diagram* if and only if all the spiders have exactly one foot. The diagrams in Fig. 11 are not α-diagrams but those in Fig. 10 are α-diagrams.

Example 5.4. In Fig. 14 the diagram d_1 is semantically equivalent to the α-diagram $d_2 \vee d_3 \vee d_4 \vee d_5$. That is, all the models for d_1 are models for $d_2 \vee d_3 \vee d_4 \vee d_5$ and vice versa.

Theorem 5.1. *Every spider diagram with constants is semantically equivalent to an α-diagram with constants.*

Proof (*Sketch*). Spider legs represent disjunction within a unitary diagram, d. Therefore, if there is a spider, s, in d that inhabits region $r_1 \cup r_2$ where $r_1 \cap r_2 = \emptyset$ then d is semantically equivalent to $d_1 \vee d_2$ where each of d_1 and d_2 are copies of d except that s inhabits r_1 in d_1 and r_2 in d_2, thus removing a spider's leg. This process of *splitting spiders* can be repeated until all spiders inhabit exactly one zone. □

Thus, for each α-diagram with constants if we can find an expressively equivalent spider diagram without constants then we will have shown that augmenting the language of

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

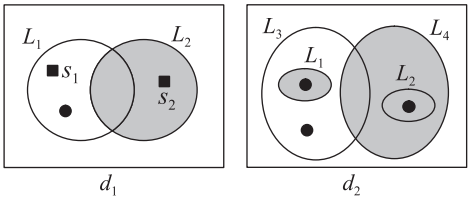

Fig. 15. Changes in the zone set.

spider diagrams with constants does not increase expressiveness. To begin, we consider unitary α-diagrams.

Example 5.5. The diagrams in Fig. 15 are expressively equivalent, given $\mathscr{CS} = \{s_1, s_2\}$. By relabelling the contours in d_1 when constructing d_2, we have changed the zone set. More drastic, though, are changes to the zone set that occur when replacing each constant spider by a contour (along with the shading and an existential spider). The zones in d_1 are

$$Z(d_1) = \{(\{L_1\}, \{L_2\}), (\{L_1, L_2\}, \emptyset), (\{L_2\}, \{L_1\}), (\emptyset, \{L_1, L_2\})\}.$$

Each of these zones gives rise to a zone in d_2, for example $z_1 = (\{L_1, L_2\}, \emptyset)$ gives rise to $z_2 = (\{L_3, L_4\}, \{L_1, L_2\})$. We have used the containing label set for z_1, namely $\{L_1, L_2\}$ and applied $\mathscr{L}$ to each of its elements to give the containing label set for z_2, namely $\{L_3, L_4\}$. Since the contour label set for d_2 is generated from the contour label set and constant spider label set in d_1 we can deduce the excluding label set for z_2:

$$\{L_1, L_2\} = L(d_2) - \{L_3, L_4\}.$$

If a zone is shaded in d_1 then it gives rise to a shaded zone in d_2. Moreover, if an existential spider inhabits z in d_1 then it inhabits the zone that z gives rise to in d_2. This establishes the habitat for each existential spider in d_2 that arises from an existential spider in d_1. Further zones, all of which are shaded, are in d_2; there is one such zone for each constant spider. As an example, the constant spider s_1 gives rise to the zone $z_3 = (\{L_1, L_3\}, \{L_2, L_4\})$. In d_1, spider s_1 has habitat $z_4 = (\{L_1\}, \{L_2\})$. The constant spider s_2 gives rise to the shaded zone $(\{L_2, L_4\}, \{L_1, L_3\})$.

In building the translation, we need to identify which constant spiders are joined by ties. The constant elimination collapses the constant spiders joined by ties into a single existential spider. Recall that θ_d is a function that returns, for each constant spider (label) in unitary diagram d, the region which in which that constant spider is placed (its habitat).

Definition 5.5. We define, for unitary α-diagram d, the set of *connected constant spider components*, denoted $ConS(d)$, to be

$$ConS(d) = \{[s_i] : s_i \in CS(d)\},$$

where $[s_i]$ is the equivalence class $[s_i] = \{s_j \in CS(d) : \omega_d(s_i, s_j) \neq \emptyset\}$. We denote the set of connected constant spider components in a zone z of d by $ConS(z, d) = \{[s_i] \in ConS(d) : \theta_d(s_i) = \{z\}\}$.

For example, in Fig. 11, we have the equivalence classes $[s_1] = \{s_1\}$, $[s_2] = \{s_2\}$ and $[s_3] = \{s_3, s_4\}$ and $ConS(d_1) = \{[s_1], [s_2], [s_3]\}$. Also, we have $ConS((\{L_1\}, \{L_2\}), d_1) = \{[s_1], [s_2]\}$.

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

For the next step in our translation, we identify the contour labels and the zones that an expressively equivalent diagram must have.

Definition 5.6. Let d be a unitary α-diagram with constants. First, we define the set of contour labels that arise from the contour labels in d, which we call $OldL(d)$. The contour labels in $OldL(d)$ are generated by applying $\mathscr{L}$ to the contour labels in $L(d)$. More formally,

$$OldL(d) = \{\mathscr{L}(L_i) : L_i \in L(d)\}.$$

Further contour labels are generated from the constant spiders in d by applying $\mathscr{L}$ to the constant spider labels giving a set we call $NewL(d)$. More formally,

$$NewL(d) = \{\mathscr{L}(s_i) : s_i \in CS(d)\}.$$

We define the zone sets $OldZ(d)$ and $NewZ(d)$ as follows:

(1) The zone set $OldZ(d)$ is the set of zones that arises from the zones in d, given $CS(d)$:

$$OldZ(d) = \{(a, (OldL(d) \cup NewL(d)) - a) : \exists (x, y) \in Z(d)\ a = \{\mathscr{L}(L_i) : L_i \in x\}\}.$$

(2) The zone set $NewZ(d)$ is the set of zones that arises from the constant spiders in d:

$$NewZ(d) = \{(a, (OldL(d) \cup NewL(d)) - a) : \exists (x, y) \in Z(d)\ \exists [s_j] \in ConS(d)$$
$$\theta_d(s_j) = \{(x, y)\} \wedge a = \{\mathscr{L}(x_i) : x_i \in x \vee x_i \in [s_j]\}\}.$$

We also define the shaded zone set $OldZ^*(d)$ to be

$$OldZ^*(d) = \{(a, (OldL(d) \cup NewL(d)) - a) : \exists (x, y) \in Z^*(d)\ a = \{\mathscr{L}(L_i) : L_i \in x\}\}.$$

In Example 5.5, we have

(1) The set

$$OldL(d_1) = \{L_3, L_4\}$$

arises from the contour labels in d_1 and

$$NewL(d_1) = \{L_1, L_2\}$$

arise from the spider labels in d_1. The union $OldL(d_1) \cup NewL(d_1)$ is the set of contour labels in d_2.

(2) The set

$$OldZ(d_1) = \{(\{L_3\}, \{L_1, L_2, L_4\}), (\{L_4\}, \{L_1, L_2, L_3\})\}$$

arises from the zone set of d_1. Arising from the constant spiders is the set of zones

$$NewZ(d_1) = \{(\{L_1, L_3\}, \{L_2, L_4\}), (\{L_2, L_4\}, \{L_1, L_3\})\}.$$

The union of these two sets, $OldZ(d_1) \cup NewZ(d_1)$, is the zone set for d_2.

(3) Finally, the set

$$OldZ^*(d_1) = \{(\{L_4\}, \{L_1, L_2, L_3\})\}$$

arises from the shaded zone set of d_1. The union $OldZ^*(d_1) \cup NewZ(d_1)$ gives the shaded zones of d_2.

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

Now we consider the existential spiders. When translating unitary diagrams, we change the contour label set. Consequently, the spider habitats also change; the next definition identifies the new habitats by way of the spider descriptors.

Definition 5.7. Let d be a unitary α-diagram with constants. We define the sets $OldE(d)$ and $NewE(d)$ as follows.

(1) The set of existential spider descriptors, $OldE(d)$, arises from the existential spider descriptors in d:

$$OldE(d) = \{(n, \{(a, (OldL(d) \cup NewL(d)) - a)\}): \exists (n, \{(x, y)\}) \in ESD(d)\ a = \{\mathscr{L}(L_i) : L_i \in x\}\}.$$

(2) The set of existential spider descriptors, $NewE(d)$, arises from the constant spiders in d:

$$NewE(d) = \{(1, \{(a, (OldL(d_1) \cup NewL(d_1)) - a)\}) : \exists (x, y) \in Z(d)\ \exists [s_j] \in ConS(d)\ \theta_d(s_j) = \{(x, y)\} \wedge a = \{\mathscr{L}(x_i) : x_i \in x \vee x_i \in [s_j]\}\}.$$

In Example 5.5, we have

$$OldE(d_1) = \{(1, \{(\{L_3\}, \{L_1, L_2, L_4\})\})\}$$

and

$$NewE(d_1) = \{(1, \{(\{L_1, L_3\}, \{L_2, L_4\})\}), (1, \{(\{L_2, L_4\}, \{L_1, L_3\})\})\}.$$

The union $OldE(d_1) \cup NewE(d_1)$ is the set of existential spider descriptors for d_2.

Definition 5.8. We define $\mathscr{C}_u^\alpha$ ($\mathscr{D}_u^\alpha$) to be the set of all unitary α-diagrams with constants (unitary α-diagrams without constants). We also define $\mathscr{E}: \mathscr{C}_u^\alpha \to \mathscr{D}_u^\alpha$ to be $\mathscr{E}(d_1) = d_2$ if and only if the following all hold.

(1) The labels in d_2 are the images of the labels in d_1 under $\mathscr{L}$:

$$L(d_2) = OldL(d_1) \cup NewL(d_1).$$

(2) The zones are 'preserved' and one new zone is introduced for each connected constant spider component:

$$Z(d_2) = OldZ(d_1) \cup NewZ(d_1).$$

(3) The shaded zones are 'preserved' and one new shaded zone is introduced for each constant spider:

$$Z^*(d_2) = OldZ^*(d_1) \cup NewZ(d_1).$$

(4) The existential spiders are 'preserved' and one new existential spider is introduced for each connected constant spider component:

$$ESD(d_2) = OldE(d_1) \cup NewE(d_1).$$

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

We have translated a diagram with constants, d_1, into a diagram without constants, $\mathscr{E}(d_1)$. If the diagram d_1 has only non-empty models then $\mathscr{E}(d_1)$ is not necessarily expressively equivalent to d_1 since, in any model for d_1, the constant spider labels in $\mathscr{CS}$ all map to single element sets but it need not be the case that all the contour labels in the set $\{L_i \in \mathscr{CL} : s_i \in \mathscr{CS} - CS(d_1)\}$ map to single element sets. We take $\mathscr{E}(d_1)$ in conjunction with one unitary diagram for each constant spider label in the set $\mathscr{CS} - CS(d_1)$ which we call s_i-*constrainers*, defined as follows.

Definition 5.9. Let d_1 be a unitary diagram with constants. Let $s_i \in \mathscr{CS} - CS(d_1)$. The diagram d_2 whose component parts are as follows is called an s_i-*constrainer* for d_1, denoted $d_1 \mapsto_{s_i} d_2$.

(1) The only contour label in d_2 arises from the constant spider label s_i:

$$L(d_2) = \{L_i\}.$$

(2) The diagram d_2 is in Venn form:

$$Z(d_2) = \{(\{L_i\}, \emptyset), (\emptyset, \{L_i\})\}.$$

(3) The only shaded zone is that inside L_i:

$$Z^*(d_2) = \{(\{L_i\}, \emptyset)\}.$$

(4) There is a single existential spider in d_2, inside L_i:

$$ESD(d_2) = \{(1, \{(\{L_i\}, \emptyset)\})\}.$$

We define $\mathscr{DIAG}(d_1) = \{d_2 : \mathscr{E}(d_1) = d_2 \vee \exists s_i \in \mathscr{CS} - CS(d_1)\ d_1 \mapsto_{s_i} d_2\}$.

In Fig. 13, d_3 is an s_3-constrainer for d_1.

So far, all of the examples we have considered to illustrate the expressively equivalent relation have included spiders in the diagram to be translated. We now consider an example where we do not include any spiders in the diagram to be translated.

Example 5.6. For this example we fix the constant spider label set to be $\mathscr{CS} = \{s_1\}$. In Fig. 16, the diagram d_1 has an empty model as well as non-empty models. We must consider these two possibilities when constructing a diagram expressively equivalent to d_1. If $m = (U, \Psi)$ is a model for d_1 and $U \neq \emptyset$ then the constant spider s_1 represents a specific individual. In the translation, this is captured by diagram d_3, and m is a model for $d_2 \wedge d_3$. Alternatively, $U = \emptyset$ and in this case m is a model for d_4. The diagram d_1 is expressively equivalent to $(d_2 \wedge d_3) \vee d_4$.

We now map each α-diagram with constants but without the $\neg$ operator to an expressively equivalent diagram without constants. We denote the set of all α-diagrams with constants but without $\neg$ by $\mathscr{C}^\alpha$ and the set of all α-diagrams without constants by $\mathscr{D}^\alpha$.

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

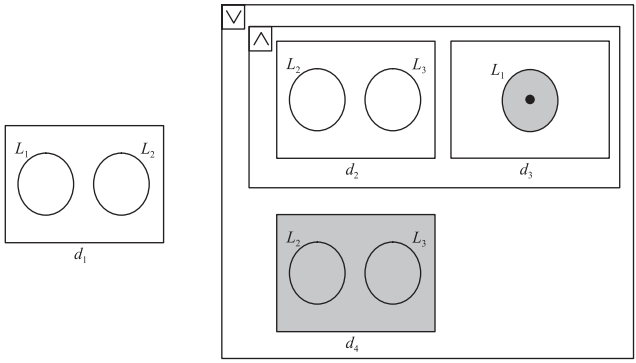

Fig. 16. Diagrams with empty models.

Definition 5.10. Define $\mathscr{EXP}: \mathscr{C}^{\alpha} \to \mathscr{D}^{\alpha}$ ($\mathscr{EXP}$ for EXPressively equivalent) as follows. Let $D \in \mathscr{C}^{\alpha}$.

(1) If D is a unitary diagram such that $S(D) \neq \emptyset$ (i.e. the set of spiders in D is not empty) then

$$\mathscr{EXP}(D) = \bigwedge_{d_2 \in \mathscr{DKB}(D)} d_2.$$

(2) If D is a unitary diagram such that $S(D) = \emptyset$ (i.e. D contains no spiders) and $Z(D) \neq Z^*(D)$ then

$$\mathscr{EXP}(D) = \left(\bigwedge_{d_2 \in \mathscr{DKB}(D)} d_2 \right) \vee d^*,$$

where d^* is a unitary α-diagram that satisfies $L(d^*) = L(\mathscr{E}(D))$, $Z(d^*) = Z(\mathscr{E}(D))$, $Z^*(d^*) = Z(\mathscr{E}(D))$ and $ESD(d^*) = ESD(\mathscr{E}(D)) = \emptyset$.

(3) If D is a unitary diagram such that $S(D) = \emptyset$ and $Z(D) = Z^*(D)$ then

$$\mathscr{EXP}(D) = \mathscr{E}(D).$$

(4) If $D = (D_1 \vee D_2)$ for some D_1 and D_2 then

$$\mathscr{EXP}(D) = (\mathscr{EXP}(D_1) \vee \mathscr{EXP}(D_2)).$$

(5) Otherwise, $D = (D_1 \wedge D_2)$ for some D_1 and D_2 and we define

$$\mathscr{EXP}(D) = (\mathscr{EXP}(D_1) \wedge \mathscr{EXP}(D_2)).$$

Theorem 5.2. *Let d_1 be a unitary α-diagram with constants. Then d_1 is expressively equivalent to $\mathscr{EXP}(d_1)$.*

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

Proof. There are three cases to consider, corresponding to the definition of $\mathscr{EXP}$ in the unitary case.

Case 1: $S(D)\neq\emptyset$. Let $m=(U,\Psi)$ be an interpretation with constants and suppose m is a model for d_1. Firstly, we will show that $h(U,\Psi)=(U,\Phi)$ is a model for $\mathscr{E}(d_1)=d_2$. To do so, we consider each of the zones in $Z(d_2)$ in turn. Let $z=(a,L(d_2)-a)\in Z(d_2)$. We will show that there exists an injective map from $\Psi(z)$ to $ES(\{z\},d_2)$ which is bijective when z is shaded in d_2. We start by noting

$$\begin{aligned}\Phi(z)&=\bigcap_{L_i\in a}\Phi(L_i)\cap\bigcap_{L_i\in L(d_2)-a}\overline{\Phi(L_i)}\\&=\bigcap_{L_i\in a}\Psi(\mathscr{L}^{-1}(L_i))\cap\bigcap_{L_i\in L(d_2)-a}\overline{\Psi(\mathscr{L}^{-1}(L_i))}\\&=\bigcap_{\mathscr{L}(P_i)\in a}\Psi(P_i)\cap\bigcap_{\mathscr{L}(P_i)\in L(d_2)-a}\overline{\Psi(P_i)}.\end{aligned}\tag{1}$$

First, we consider the subcase where a contains a contour label, L_j say, where the subscript, j, satisfies $j\leqslant n$. That is, the zone $z=(a,b)$ arose from a constant spider and is in the set $NewZ(d_1)$. In this subcase, z is shaded in d_2 and contains exactly one existential spider. We show that $\Phi(z)=\Psi(s_j)$, which allows us to deduce $|\Phi(z)|=1$ and, hence, the required bijection exists. By (1) and since $\mathscr{L}(s_j)=L_j\in a$,

$$\Phi(z)=\bigcap_{\mathscr{L}(P_i)\in a}\Psi(P_i)\cap\bigcap_{\mathscr{L}(P_i)\in L(d_2)-a}\overline{\Psi(P_i)}\cap\Psi(s_j).$$

We have

$$a=\{\mathscr{L}(x_i):x_i\in x\vee x_i\in[s_j]\},$$

where $\theta_{d_1}(s_j)=\{(x,y)\}$ and

$$L(d_2)-a=\{\mathscr{L}(y_i):y_i\in y\vee y_i\in CS(d_1)-[s_j]\}.$$

Therefore, from (1) and since $\mathscr{L}(s_j)\in a$,

$$\begin{aligned}\Phi(z)&=\bigcap_{\mathscr{L}(P_i)\in\{\mathscr{L}(x_i):x_i\in x\}}\Psi(P_i)\cap\\&\qquad\bigcap_{\mathscr{L}(P_i)\in\{\mathscr{L}(y_i):y_i\in y\vee y_i\in CS(d_1)-[s_j]\}}\overline{\Psi(P_i)}\cap\bigcap_{s_i\in[s_j]}\Psi(s_i)\\&=\bigcap_{P_i\in x}\Psi(P_i)\cap\bigcap_{P_i\in y\cup(CS(d_1)-[s_j])}\overline{\Psi(P_i)}\cap\bigcap_{s_i\in[s_j]}\Psi(s_j)\\&=\bigcap_{P_i\in x}\Psi(P_i)\cap\bigcap_{P_i\in y}\overline{\Psi(P_i)}\cap\bigcap_{P_i\in CS(d_1)-[s_j]}\overline{\Psi(P_i)}\cap\bigcap_{s_i\in[s_j]}\Psi(s_j)\\&=\Psi(x,y)\cap\bigcap_{P_i\in CS(d_1)-[s_j]}\overline{\Psi(P_i)}\cap\bigcap_{s_i\in[s_j]}\Psi(s_j).\end{aligned}\tag{2}$$

Consider the term

$$\bigcap_{P_i\in CS(d_1)-[s_j]}\overline{\Psi(P_i)}.$$

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

Now

$$\bigcap_{P_i \in CS(d_1)-[s_j]} \overline{\Psi(P_i)} = U - \bigcup_{P_i \in CS(d_1)-[s_j]} \Psi(P_i).$$

By the constant spiders condition for d_1, all the constant spiders in the set $CS(d_1) - [s_j]$ map to individuals distinct from that represented by s_j. Moreover, for all $s_i \in [s_j]$, $\Psi(s_j) = \Psi(s_i)$. We deduce that

$$\Psi(s_j) \subseteq U - \bigcup_{P_i \in CS(d_1)-[s_j]} \Psi(P_i).$$

Therefore (2) becomes

$$\Phi(z) = \Psi(x, y) \cap \Psi(s_j).$$

Now $\Psi(s_j) \subseteq \Psi(x, y)$ by the spiders condition for d_1. Therefore

$$\Phi(z) = \Psi(s_j). \tag{3}$$

Hence $|\Phi(z)| = 1 = |ES(\{z\}, d_2)| = |ET(\{z\}, d_2)|$.

Alternatively a does not contain any contour label, L_j, with subscript j that satisfies $j \leqslant n$. That is, the zone $z = (a, b)$ is in the set $OldZ(d_1)$. In this case there is a zone $(x, y) \in Z(d_1)$ such that

$$a = \{\mathscr{L}(L_i) : L_i \in x\}$$

and

$$L(d_2) - a = \{\mathscr{L}(y_i) : y_i \in y \vee y_i \in CS(d_1)\}.$$

Now, from (1),

$$\begin{aligned}\Phi(z) &= \bigcap_{L_i \in x} \Psi(L_i) \cap \bigcap_{y_i \in y \ \cup \ CS(d_1)} \overline{\Psi(y_i)} \\ &= \Psi(x, y) \cap \bigcap_{s_i \in CS(d_1)} \overline{\Psi(s_i)}.\end{aligned}$$

By the spiders condition for d_1, we deduce the following:

(a) $$\Phi(z) = \Psi(x, y) - \bigcup_{s_i \in CS(\{(x,y)\}, d_1)} \Psi(s_i). \tag{4}$$

(b) $$\bigcup_{e \in ES(\{(x,y)\}, d_1)} \Psi(e) \subseteq \Psi(x, y).$$

(c) The sets $\bigcup_{e \in ES(\{(x,y)\}, d_1)} \Psi(e)$ and $\bigcup_{s_i \in CS(\{(x,y)\}, d_1)} \Psi(s_i)$ are disjoint.

By the existential spiders condition for d_1,

$$\left| \bigcup_{e \in ES(\{(x,y)\}, d_1)} \Psi(e) \right| = |ES(\{(x, y)\}, d_1)|.$$

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

Hence

$$|\Phi(z)| = \left|\Psi(x,y) - \bigcup_{s_i \in CS(\{(x,y)\},d_1)} \Psi(s_i)\right| \geqslant |ES(\{(x,y)\},d_1)| = |ES(\{z\},d_2)|.$$

Thus $|\Phi(z)| \geqslant |ES(\{z\},d_2)|$. Suppose that z is shaded in d_2. Then (x,y) is shaded in d_1 and

$$\begin{aligned}|\Phi(z)| &= \left|\Psi(x,y) - \bigcup_{s_i \in CS(\{(x,y)\},d_1)} \Psi(s_i)\right| \\ &\leqslant |ET(\{(x,y)\},d_1)| - |ConS((\{(x,y)\},d_1))| \\ &= |ES(\{(x,y)\},d_1)| - |ConS((\{(x,y)\},d_1))| \\ &\leqslant |ES(\{z\},d_2)|.\end{aligned}$$

Hence $|\Phi(z)| \leqslant |ES(\{z\},d_2)|$, so $|\Phi(z)| = |ES(\{z\},d_2)|$.

We deduce that, for any $z \in Z(d_2)$, there exists an injective map from $ES(\{z\},d_2)$ to $\Phi(z)$. Moreover, when z is shaded in d_2, such an injective map is also bijective. It is, therefore, straightforward to show that there exists a valid extension of Ψ to existential spiders for d_2. By considering (3) and (4) along with the construction of $Z(d_2)$, it can be shown that, since the plane tiling condition holds for d_1, the plane tiling condition and the constant spiders condition hold for d_2. Hence $h(U,\Psi) = (U,\Phi)$ is a model for d_2. Noting that, since d_1 contains at least one spider, $U \neq \emptyset$, it is straightforward to show that (U,Φ) is a model for each diagram in the set $\mathscr{DIAG}(d_1) - \{\mathscr{E}(d_1) = d_2\}$. Therefore (U,Φ) is a model for

$$D = \bigwedge_{d \in \mathscr{DIAG}(d_1)} d.$$

What remains is to show that any model for D is the image of some model for d_1. Let (U,Φ) be a model for D. The strategy is to start by showing $h^{-1}(U,\Phi)$ is defined and then follow a similar argument to the first part of the proof.

Case 2: $S(d_1) = \emptyset$ and $Z(d_1) \neq Z^*(d_1)$. In this case, we note that in any model $m = (U,\Psi)$ for d_1, either $U = \emptyset$ or $U \neq \emptyset$. In the first subcase, m is a model for d^*. In the second subcase, m is a model for

$$\bigwedge_{d \in \mathscr{DIAG}(d_1)} d.$$

The strategy used in case 1 can be modified (and simplified) to both subcases.

Case 3: $S(d_1) = \emptyset$ and $Z(d_1) = Z^*(d_1)$. In this case, d_1 has only the empty model, and the proof is similar to the first subcase of case 2. Hence d_1 is expressively equivalent to $\mathscr{EXP}(d_1)$. □

Theorem 5.3. *Let D be an α-diagram with constants but without the $\neg$ operator. Then D is expressively equivalent to $\mathscr{EXP}(D)$.*

Proof. The proof follows by induction on the depth of D in the inductive construction, with the base case provided by Theorem 5.2. □

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

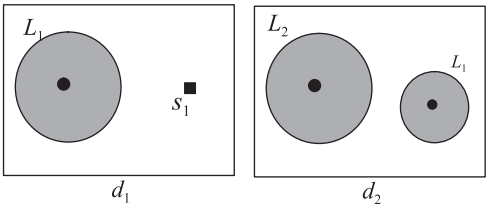

Fig. 17. Negation issues.

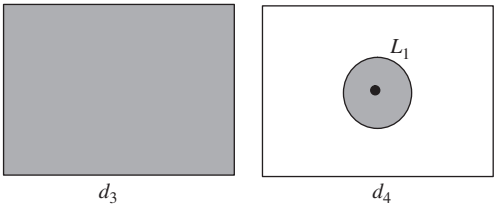

Fig. 18. Fixing negation issues.

5.1. Incorporating negation

Our focus now turns to the case where we allow the $\neg$ operator to be used. First, we illustrate why extending the translation to include negation is not as straightforward as for $\wedge$ and $\vee$.

Example 5.7. Taking $\mathscr{CS} = \{s_1\}$, the unitary diagram d_1 in Fig. 17 is expressively equivalent to d_2. However, $\neg d_1$ is not equivalent to $\neg d_2$. For example, the interpretation without constants $m = (\{1,2\}, \Psi)$, where $\Psi(L_1) = \{1,2\}$ and $\Psi(L_2) = \emptyset$, is a model for $\neg d_2$ but not in the image of h used to define the expressively equivalent relation (s_1 maps to L_1, but s_1 never represents a set containing two elements). In other words, h does not provide a bijective correspondence between the models for $\neg d_1$ and $\neg d_2$.

Thus, the 'problem' is that all interpretations with constants force the constant spider labels to denote single element sets or the empty set (when $U = \emptyset$). So, when we translate the negation of a unitary diagram, we must ensure that the contour labels arising from the constant spider labels (under $\mathscr{L}$) also have this property.

Example 5.8. Returning to Fig. 17, we observe that $\neg d_1$ is semantically equivalent to $\neg d_2 \wedge (d_3 \vee d_4)$ where d_3 and d_4 are in Fig. 18.

In general, an extension of the definition of $\mathscr{EXP}$ to each diagram with constants of the form $\neg D$ is[6]

$$\mathscr{EXP}(\neg D) = \neg \mathscr{EXP}(D) \wedge \left(d_3 \vee \bigwedge_{d \in \mathscr{ZMB}(\square)} d \right);$$

[6]We use $\square$ to represent the unitary diagram containing no contours, no spiders and no shading.

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

the conjunction

$$\bigwedge_{d \in \mathscr{DIAG}(\Box)} d$$

expresses that each of the contour labels arising from the constant spider labels in $\mathscr{CS}$ represent single element sets whereas d_3 expresses the fact that the universe is empty. The diagram $\neg D$ is expressively equivalent to $\mathscr{EXP}(\neg D)$.

The definition of $\mathscr{EXP}(\neg D)$ returns a spider diagram without constants that includes negation. However, it immediately follows from the expressiveness result in [11] that negation is a derived operator in this language. Thus we have the following result.

Theorem 5.4. *Augmenting the spider diagram language with constants, ties and negation does not increase expressiveness.*

Proof. We must show that every diagram with constants has an expressively equivalent diagram without constants. We have shown that for every α-diagram with constants there exists an expressively equivalent diagram without constants. Let D_1 be a diagram with constants. By Theorem 5.1, D_1 is semantically equivalent to some α-diagram with constants, D_2 say. Since D_1 and D_2 have the same models, it follows that h provides a bijective correspondence between the models for D_1 and those for $\mathscr{EXP}(D_2)$. Therefore D_1 is expressively equivalent to the diagram without constants $\mathscr{EXP}(D_2)$. Hence augmenting the spider diagram language with constants does not increase expressiveness. □

We proved in [11], that the language of spider diagrams without constants is equivalent in expressive power to monadic first order logic with equality. Hence we obtain the following corollary.

Corollary 5.1. *The language of spider diagrams with constants is equivalent in expressive power to monadic first order logic with equality.*

5.2. Further discussion

We mentioned in Section 3 that there are two possible ways of performing the construction which eliminates constant spiders. The construction given above works with a finite set of constant spider labels; an alternative approach would use an infinite set of constant spider labels. The advantage of using this alternative approach is that we can always be sure of having enough constant spider labels for our purposes; the disadvantage is that the construction becomes infinite in some places.

In essence, the change comes from the fact that the set of diagrams, $\mathscr{DIAG}(d)$, becomes infinite rather than finite, since one diagram is required for each constant spider label. We now discuss the effect of defining $\mathscr{CS}$ to be a countably infinite set of constant spider labels, rather than a finite set at a more detailed level. We redefine $\mathscr{L}: \mathscr{CS} \cup \mathscr{CL} \to \mathscr{CL}$ by

$$\mathscr{L}(x_i) = \begin{cases} x_{2i} & \text{if } x_i \in \mathscr{CL}, \\ x_{2i-1} & \text{if } x_i \in \mathscr{CS}. \end{cases}$$

If we use the same translation mapping, $\mathscr{EXP}$, with an infinite set of constant spider labels then some unitary diagrams, d, map to infinite conjunctions of diagrams since the set

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

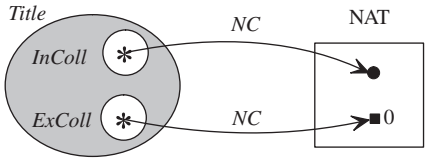

Fig. 19. A constraint diagram with a constant.

$\mathscr{DIAG}(d)$ is not necessarily finite. So, rather than $\mathscr{EXP}(d)$ returning a diagram, the function would need to be redefined to return a (sometimes infinite) set of diagrams representing an infinite conjunction. This set of diagrams is expressively equivalent to d. In the language of spider diagrams with constants (with an infinite set of constant spider labels) the property that all the constant spider labels represent individuals is finitely axiomatizable (for example, by a unitary diagram containing exactly one spider). By contrast, in the language of spider diagrams without constants, the property that all the contour labels which arise from constant spider labels (of which there are infinitely many) represent single element sets is infinitely axiomatizable (for each such contour label L_i, the unitary diagram that contains L_i with a single existential spider and shading is an axiom) but not finitely.

6. Conclusion

In this paper, we have augmented the spider diagram language with constants and provided a formalization of the extended system. Subsequently, we proved that this augmentation does not lead to an increase in expressive power. However, we believe that if one wishes to make statements about specific individuals then it is natural to do so using constants rather than a contour, shading and an existential spider. Thus augmenting with constants, although it brings no expressiveness benefits, is highly likely to increase the usability of the notation. For this reason, the formalization of constraint diagrams, found in [8], would benefit from being extended to include constants.

As an example, the constraint diagram in Fig. 19 is taken from the specification of a library system and indicates that titles (in the library catalogue) have a 'number of copies' (*NC*) associated with them (NAT is the set of *natural numbers*, which includes 0). Titles are partitioned into two disjoint subsets: those having no copies are said to be 'ex-collection' (*ExColl*); all others are 'in-collection' (*InColl*).

The highly expressive nature of constraint diagrams may mean that including constants in the language brings even more usability benefits: when making complex statements, it should not be overly difficult to find a diagram capturing the required constraint.

The approach we have taken to eliminate constants is likely to generalize to other languages. Consequently, this work is of greater significance than the face value of the results alone. Furthermore, a modification of the high level approach and the constant elimination strategy can be used to compare the expressiveness of syntactically disparate languages.

References

[1] S.-J. Shin, The Logical Status of Diagrams, Cambridge University Press, Cambridge, 1994.
[2] J. Larkin, H. Simon, Why a diagram is (sometimes) worth ten thousand words, Journal of Cognitive Science 11 (1987) 65–99.

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005

[3] Unified Modeling Language ⟨http://www.uml.org/⟩, 2006.
[4] S. Kent, Constraint diagrams: visualising invariants in object oriented modelling, in: OOPLSA 97, ACM SIGPLAN Notices 32(10) (1997) 327–341.
[5] S. Kent, Three dimensional software modelling, Proceedings of the International Conference on Software Engineering, IEEE, 1998, pp. 105–114.
[6] J. Howse, S. Schuman, Precise visual modelling, Journal of Software and Systems Modeling 4 (2005) 310–325.
[7] S.-K. Kim, D. Carrington, Visualization of formal specifications, in: 6th Aisa Pacific Software Engineering Conference, IEEE Computer Society Press, Los Alamitos, CA, USA, 1999, pp. 102–109.
[8] A. Fish, J. Flower, J. Howse, The semantics of augmented constraint diagrams, Journal of Visual Languages and Computing 16 (2005) 541–573.
[9] J. Howse, G. Stapleton, J. Taylor, Spider diagrams, LMS Journal of Computation and Mathematics 8 (2005) 145–194.
[10] F. Molina, Reasoning with extended Venn–Peirce diagrammatic systems, Ph.D. Thesis, University of Brighton, 2001.
[11] G. Stapleton, S. Thompson, J. Howse, J. Taylor, The expressiveness of spider diagrams, Journal of Logic and Computation 14 (6) (2004) 857–880.
[12] J. Howse, F. Molina, J. Taylor, S. Kent, J. Gil, Spider diagrams: a diagrammatic reasoning system, Journal of Visual Languages and Computing 12 (3) (2001) 299–324.
[13] N. Swoboda, G. Allwein, Using DAG transformations to verify Euler/Venn homogeneous and Euler/Venn FOL heterogeneous rules of interence, Journal on Software and System Modeling 3 (2) (2004) 136–149.
[14] N. Swoboda, Implementing Euler/Venn reasoning systems, in: M. Anderson, B. Meyer, P. Olivier (Eds.), Diagrammatic Representation and Reasoning, Springer, Berlin, 2001, pp. 371–386.
[15] G. Stapleton, J. Howse, J. Taylor, S. Thompson, What can spider diagrams say?, in: Proceedings of Diagrams 2004, Springer, Berlin, 2004, pp. 112–127.
[16] G. Stapleton, J. Howse, J. Taylor, S. Thompson, The expressiveness of spider diagrams augmented with constants, in: Proceedings of the Visual Languages and Human Centric Computing, Rome, Italy, IEEE Computer Society Press, Silver Spring, MD, September 2004, pp. 91–98.
[17] R. Clark, Failure mode modular de-composition using spider diagrams, in: Proceedings of Euler Diagrams 2004, Electronic Notes in Theoretical Computer Science, vol. 134, 2005, pp. 19–31.
[18] J. Thievre, M.-L. Viaud, A. Verroust-Blonde, Using Euler diagrams in traditional library environments, in: Euler Diagrams 2004, Electronic Notes in Theoretical Computer Science, vol. 134, Elsevier, Amsterdam, 2005.
[19] R. DeChiara, U. Erra, V. Scarano, VennFS: a Venn diagram file manager, in: Proceedings of Information Visualisation, IEEE Computer Society, Silver Spring, MD, 2003, pp. 120–126.
[20] J. Lövdahl, Towards a visual editing environment for the languages of the semantic web, Ph.D. Thesis, Linkoping University, 2002.
[21] Y. Zhao, J. Lövdahl, A reuse based method of developing the ontology for e-procurement, in: Proceedings of the Nordic Confernce on Web Services, 2003.
[22] P. Hayes, T. Eskridge, R. Saavedra, T. Reichherzer, M. Mehrotra, D. Bobrovnikoff, Collaborative knowledge capture in ontologies, in: Proceedings of the 3rd International Conference on Knowledge Capture, 2005, pp. 99–106.
[23] A. Fish, G. Stapleton, Formal issues in languages based on closed curves, in: Proceedings of Distributed Multimedia Systems, International Workshop on Visual Languages and Computings, Knowledge Systems Institute, Grand Canyon, USA, 2006, pp. 161–167.
[24] J. Howse, F. Molina, S.-J. Shin, J. Taylor, On diagram tokens and types, in: Proceedings of Diagrams 2002, Springer, Berlin, 2002, pp. 76–90.
[25] P.T. Johnstone, Notes on Logic and Set Theory, CUP, 1987.
[26] G. Stapleton, Reasoning with constraint diagrams, Ph.D. Thesis, University of Brighton (August 2004), published by the British Computer Society ⟨www.bcs.org/server.php?show=conWebDoc.2842⟩, 2005.

Please cite this article as: G. Stapleton, et al., The expressiveness of spider diagrams augmented with constants, Journal of Visual Language and Computing (2008), doi:10.1016/j.jvlc.2008.01.005