Euler Diagrams 2012

Chapman, Micallef (eds.)

2012

Proceedings of the 3rd International Workshop on Euler Diagrams

Euler Diagrams 2012

2nd July 2012 Canterbury, UK

Peter Chapman
Luana Micallef
(editors)

Proceedings of the

3rd International Workshop on Euler Diagrams

Euler Diagrams 2012

2nd July 2012, Canterbury, UK

in conjunction with Diagrams 2012

Proceedings of the
3rd International Workshop on Euler Diagrams

Euler Diagrams 2012

Peter Chapman
Luana Micallef
(editors)

Euler Diagrams

3rd International Workshop, Euler Diagrams 2012

Canterbury, UK, July 2, 2012

Proceedings

Also online at CEUR-WS.org/Vol-854

CEUR Workshop Proceedings
http://ceur-ws.org
ISSN 1613-0073

Editors:

Peter Chapman
University of Brighton, UK
P.B.Chapman@brighton.ac.uk

Luana Micallef
University of Kent, UK
lm304@kent.ac.uk

Preface

The 3rd International Workshop on Euler Diagrams (Euler Diagrams 2012) was held in Canterbury, UK on 2nd July 2012 in conjunction with the 7th International Conference on the Theory and Application of Diagrams (Diagrams 2012).

Euler diagrams represent relationships between sets, including intersection, containment, and disjointness. These diagrams have become the foundations of various visual languages and have notably facilitated the modelling of, and logical reasoning about, complex systems. Over the years, they have been used extensively in areas such as biosciences, business, criminology and national security to intuitively visualize relationships and relative cardinalities of sets. This widespread adoption has allowed analysis of complex collections of data.

Euler Diagrams 2012 covered all aspects of Euler diagram research, particularly in areas such as logic and reasoning, drawability, generation and layouts, readability and aesthetics, information visualization and data exploration, and evaluation including comparison to other representations. After two successful workshops in 2004 and 2005, this third Euler Diagrams workshop once again brought together researchers with diverse backgrounds. Participants from both academia and industry included: mathematicians; computer scientists; experts in visualization, human-computer interaction and artificial intelligence; information designers; and users from various application areas.

Euler Diagrams 2012 solicited long and short papers, of which we accepted eight long papers and one short paper. Every submission was reviewed by three members of the Program Committee who are experts in the relevant topics. In addition to the paper presentations, we were privileged to have Tim Dwyer from Microsoft Corportion, USA to give a keynote talk on "Developing a Visual Code-dependency Analysis Tool for the Visual Studio IDE: Research Meets Practice in Showing Containment in an Interactive Diagramming Tool".

We owe thanks to many people for helping to make Euler Diagrams 2012 a success. First and foremost, we are grateful to our Program Committee consisting of twelve distinguished experts from both academia and industry. Their insightful reviews provided invaluable feedback to authors of submitted papers. David Dailey, John Howse, Nathaniel Miller and Nik Swoboda kindly agreed to act as Session Chairs. As part of the Organizing Committee of Diagrams 2012, we are indebted to the General Chair, Peter Rodgers, the Workshop Chair, Nathaniel Miller, and the Publicity Chair, Aidan Delaney, for making the process of organizing the workshop as smooth as possible. We also thank Gem Stapleton for her indispensable advice. Finally, we acknowledge the US National Science Foundation for funding many of the PhD students to attend Diagrams 2012, and hence, to also attend our workshop.

June 2012

Peter Chapman
Luana Micallef

Organization

General Chairs

Peter Chapman	University of Brighton, UK
Luana Micallef	University of Kent, UK

Publicity Chair

Aidan Delaney	University of Brighton, UK

Session Chairs

David Dailey	Slippery Rock University, USA
John Howse	University of Brighton, UK
Nathaniel Miller	University of Northern Colorado, USA
Nik Swoboda	Universidad Politécnica de Madrid, Spain

Program Committee

Rosario de Chiara	Università degli Studi di Salerno, Italy
Gennaro Cordasco	Seconda Università degli Studi di Napoli, Italy
Tim Dwyer	Microsoft Corportion, USA
Mateja Jamnik	University of Cambridge, UK
Stephen North	AT&T Research, USA
Mitsuhiro Okada	Keio University, Japan
Nathalie Henry Riche	Microsoft Research, USA
Peter Rodgers	University of Kent, UK
Frank Ruskey	University of Victoria, Canada
Paolo Simonetto	INRIA, France
Bettina Speckmann	TU Eindhoven, Netherlands
Gem Stapleton	University of Brighton, UK

Additional Reviewers

Matej Urbas	University of Cambridge, UK

Table of Contents

Developing a Visual Code-dependency Analysis Tool for the Visual Studio IDE: Research Meets Practice in Showing Containment in an Interactive Diagramming Tool

Tim Dwyer

Microsoft Corp., One Microsoft Way, Redmond, WA, USA
timdwyer@microsoft.com

The Microsoft Visual Studio IDE includes various tools for diagrammatic code understanding. I am currently involved in the final stages of preparing the next major version (11) for release later this year. In particular, I am working on a visual dependency analyzer that enables developers to dynamically build up a diagram from a particular piece of code or functionality. In designing and developing this product we have explored a number of methods for showing containment of various code elements to different types of grouping. Such a grouping—for example, members within types, types within namespaces, and so on—is fundamental to organizing object-oriented software, but is also the ticket to scalability of diagrams representing code. That is, grouping at various levels provides degrees of abstraction that can be applied to reduce the complexity of the visualization.

Using the Euler diagramming convention to show these groupings as overlapping regions seems very natural. However, as many researchers in the field have observed, the topology of these overlapping regions can quickly become complex and extremely difficult to draw in a readable way. In our exploration of the design space for visual code understanding we have experimented with many different ways to effectively convey grouping in code-dependency diagrams. It is a cross-disciplinary effort involving developers and UX researchers from within the Visual Studio team, in collaboration with HCI and algorithms researchers from Microsoft Research. Some of this work has been published as research papers while some of it has given us valuable insight but has not yet been distilled into easily publishable units. In this talk, I look forward to sharing some of these anecdotes as I reminisce the experience of developing a high-quality commercial domain-specific diagramming tool for mainstream customers. A process that has also necessitated exploration of novel visual conventions, layout techniques and interaction.

Completeness Proof Strategies for Euler Diagram Logics

Jim Burton, Gem Stapleton, and John Howse

Visual Modelling Group, University of Brighton, UK
{j.burton,g.e.stapleton,john.howse}@brighton.ac.uk

Abstract. Visual logics based on Euler diagrams have recently been developed, including generalized constraint diagrams and concept diagrams. Establishing the metatheories of these logics includes providing completeness proofs where possible. Completeness has been established for such logics, including Euler diagrams, spider diagrams and a fragment of the constraint diagram logic. In this paper, we identify commonality in their completeness proof strategies, showing how, as expressiveness increases, the strategy readily extends. We identify a fragment of concept diagrams and demonstrate that the completeness proof strategy does not extend to this fragment. Thus, we have established that the existing completeness proof strategies are limited. Consequently, we examine the challenge of devising new approaches to proving completeness in more expressive logics.

1 Introduction

There has been a lot of recent interest in logics that, in various ways, extend Euler diagrams. This interest was sparked by pioneering work in the mid 1990s, by Hammer [3] and Shin [9]. Hammer developed a very simple sound and complete Euler diagram logic, whereas Shin devised a logic, called Venn-II, that was more expressive than Euler diagrams and which she also proved to be sound and complete. Since these early days we have seen the development of diagrammatic logics with ever-increasing levels of expressiveness. Amongst these logics, perhaps the most studied is that of spider diagrams, introduced by Gil et al. [2], which arose from Kent's constraint diagram logic [6], formalised in [1]. Building on from the complete systems of Hammer and Shin, spider diagrams have been shown to be complete [4], as has a fragment of the constraint diagram logic [11]. Other related logics include the Euler/Venn system of Swoboda and Allwein [13] and the Euler system of Mineshima et al. [7].

One reason that significant emphasis has been placed on deriving completeness results for logics is that completeness means that the logic is capable of proving all theorems expressible within the logic. Formally, a theorem is a statement that semantically follows from a set of statements formulated in the logic, called axioms. In the case of diagrammatic logics, the set of axioms is a set of diagrams and a theorem is a diagram whose informational content is derivable from the axioms. For a theorem to be provable from the axioms, we need to be able to apply so-called *inference rules*, which are (informally) transformations that alter the syntax of the axioms, until we obtain the theorem.

This paper has two key parts. First, in section 2, we will demonstrate that there are substantial similarities in existing completeness proof strategies for Euler-based diagrammatic logics, with the result that we can consider the strategies to be variations on a single approach. In section 3 we describe the task of extending the proof strategy to a fragment of concept diagrams and show that the strategy breaks down. We examine the factors whose interaction prevents the ready extension of the strategy and show that completeness proofs for more expressive notations will require a different approach. We conclude in section 4 by describing some of the approaches that may be taken to finding suitable new strategies.

2 Completeness Strategies for Euler Diagram Logics

There have been a number of sound and complete logics based on Euler diagrams developed to date. All of the proofs of completeness have used constructive strategies, providing a proof that the theorem follows from the axioms (in fact, those strategies we demonstrate are restricted to a single axiom). Moreover, they all adopt a similar framework, converting the diagrams involved into normal forms that are easily comparable. As we shall demonstrate in this section, the completeness proof for each considered logic is an extension of the completeness proofs for its fragments. We show this by detailing the strategies used for a hierarchy of increasingly expressive logics: Euler diagrams [3], spider diagrams [4] and, briefly, constraint diagrams as considered in [11].

2.1 Euler Diagrams

Euler diagrams, as investigated by Hammer [3], are the simplest logic that we will consider. They comprise closed curves, each with a label. In any given diagram, no two distinct curves have the same label. Examples can be seen in figure 1, where d expresses that (the sets) A and C are disjoint, B is a subset of A, and D is a subset of C. The diagram d' expresses that D is a subset of C. Whilst the curves (labelled) A and E are present in d', no information is given about the relationship of the sets they represent to C and D.

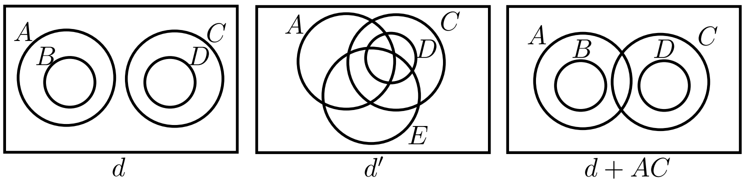


Fig. 1. Three Euler diagrams.

Hammer's logic contains just three inference rules: *Erasure* (of a curve), *Introduction of a New Curve*, and *Weakening* which allows new regions to be added; Weakening is illustrated in figure 1, where $d + AC$ is obtained from d by adding a region inside both A and C. To prove completeness of this logic, Hammer proceeds by constructing a proof-writing algorithm: given an axiom d and a theorem d', carry out the following steps to prove d' follows from d:

1. *Apply the Introduction of a New Curve rule*, adding one curve labelled L for each curve label, L, in d' that is not in d, to give a diagram d_c.
2. *Apply the Erasure rule*, erasing all curves from d_c that have labels not appearing in d', to give a diagram d_e.
3. *Apply the Weakening rule*, adding minimal regions to d_e until it is the same as d'.

The proof of completeness involves showing that it is possible to apply this algorithm whenever $d \vDash d'$ (i.e. d semantically entails d'), thus establishing that $d \vdash d'$ (i.e. there is a proof that d' follows from d). We observe that the first step of this proof can be considered as kind of *maximising* step: syntax is added to the axiom that is used in the theorem. The last step of the proof also adds syntax. In fact, we can interchange the last two steps without significantly impacting the details of the completeness proof. Thus, if we add minimal regions before erasing curves we would genuinely have *maximised* the syntax in the axiom diagram so that only inference steps that erase syntax are required in order to obtain d'. In what follows, we denote the maximised version of d by d_{max}, and we have, instead:

1. *Apply the Introduction of a New Curve rule*, adding one curve labelled L for each curve label, L, in d' that is not in d, to give a diagram d_c; we can similarly obtain d'_c, which we will use to determine inference rule applications at the next step.
2. *Apply the Weakening rule*, adding minimal regions to d_w until it has the same the same minimal regions as d'_c, to obtain $d_w = d_{max}$, the maximised version of d.
3. *Apply the Erasure rule*, erasing all curves from d_{max} that have labels not appearing in d', to give a diagram d_e. Then $d_e = d'$.

That is, we have:

$$d \vdash d_c \vdash d_w = d_{max} \vdash d_e = d'.$$

To determine which minimal regions to add to obtain d_{max}, we constructed a diagram, d'_c from d' by adding the curves with labels that occur in d but not in d'. Then d_c and d'_c have the same curve labels, so the minimal regions are immediately comparable; we add regions that are in d'_c but are not in d_c, thus maximising the syntax to get d_{max}. As we shall see, this concept of maximising the syntax in the axiom diagram is a recurring theme in subsequently developed completeness proofs. The completeness proof strategy is illustrated in figure 2, where we show $d \vdash d'$.

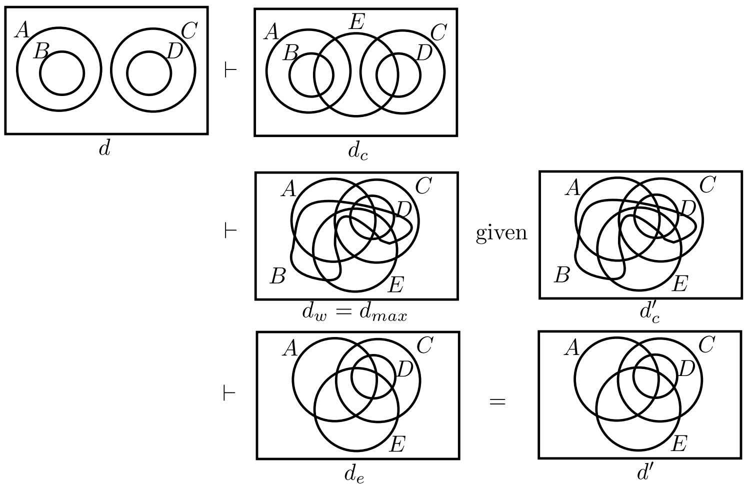


Fig. 2. Proving $d \vdash d'$.

In order to illustrate the extension of this strategy to more expressive systems, including spider diagrams in the next subsection, we formalize the notion of maximal forms. First, we define an (abstract) Euler diagram:

Definition 1. *An **Euler diagram** is a pair, $d = (L, R)$, where L is a finite set of curve labels and $R \subseteq \{(in, L - in) : in \subseteq L\}$ is a finite set of regions.*

So, d in figure 2 is, formally, $d = (L, R)$ where $L = \{A, B, C, D\}$ and

$$R = \{(\emptyset, \{A, B, C, D\}), (\{A\}, \{B, C, D\}), (\{A, B\}, \{C, D\}), (\{C\}, \{A, B, D\}), (\{C, D\}, \{A, B\})\}.$$

For example, $(\{A\}, \{B, C, D\})$ corresponds to the region inside the curve labelled A but outside the curves labelled B, C, and D[1].

Now, going back to the completeness proof strategy, we have seen that d_{max} is created by constructing the diagram d'_c which does not formally comprise part of the proof that $d \vdash d'$; in figure 2 we have $d_{max} = d'_c$ when $d \vdash d'$. We define the maximal form as follows:

Definition 2. *Let $d_c = (L, R)$ and $d'_c = (L', R')$ be Euler diagrams such that $L = L'$. The diagram d_c is **maximal** with respect to d'_c provided $R' \subseteq R$.*

It can be shown, given that d_c is maximal with respect to d'_c, $d \vDash d'_c$ if and only if $R = R'$. In terms of the completeness proof strategy, this means that $d_{max} = d'_c$. Our re-ordering of the steps in Hammer's completeness proof can now be informally justified. Firstly, to d_c we add precisely the minimal regions in d'_c that are not in d_c to give $d_w = d_{max}$. Since d'_c is semantically equivalent to d' and $d'_c = d_{max}$, it should be easy to see that we can then merely delete curves from d_{max} to give d', establishing completeness.

[1] The elements of R are often called *zones* but in this paper we call them regions for consistency with Hammer's work.

2.2 Spider Diagrams

Spider diagrams extend the Euler diagram logic of Hammer in two distinct ways: they are augmented with (a) trees, called *spiders*, and shading, both of which are used within diagrams to place constraints on set cardinality, and (b) logical connectives which are used to allow more complex expressions to be formed. Whilst Euler diagrams form a very simple monadic first-order logic, spider diagrams take the level of expressiveness to monadic first-order logic with equality [12].

Examples of spider diagrams can be seen in figure 3 where, in addition to the information provided by the underlying Euler diagram, d_1 expresses – using spiders – that there are at least two elements, one of which is in B and the other of which is in $B \cup D$. Diagram d_1 also expresses – using shading – that no further elements are in B. Here, each of the spiders (one of which comprises a single node) represents the existence of an element. The shading in a region, r, expresses that all elements in the set represented by r must be represented by spiders. The spider diagram $d_2 \vee d_3$ is semantically equivalent to d_1.

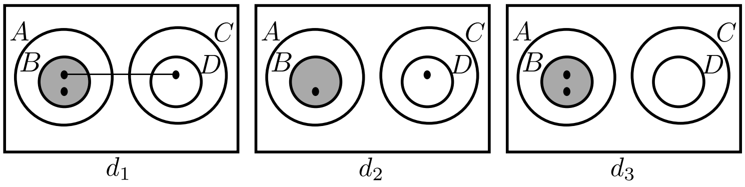


Fig. 3. Three spider diagrams.

The completeness proof strategy for spider diagrams, from [4], starts with axiom d and theorem d', so $d \vDash d'$, and, as with Hammer's approach, constructs a proof to show that $d \vdash d'$. In brief, the process starts off by converting d to a normal form where the only logical connective used is $\vee$ and the spiders each comprise just a single node, giving a diagram we will denote by d_{NF} (*NF* for Normal Form). Of note is that the construction of d_{NF} includes some of the steps we need to maximise syntax in the axiom: all of the so-called *unitary* diagrams contain all of the curve labels that occur somewhere in either the axiom or theorem. Unitary diagrams are spider diagrams which do not involve any logical connectives. In addition, the unitary diagrams in this normal form contain the same sets of regions. For Euler diagrams we constructed d'_c to direct which regions we needed to add. The same approach is used for spider diagrams: we convert diagram d' to, in this case, d'_{NF} in order to allow us to identify which inference rules to apply to d_{NF} to give d'_{NF} and, subsequently, to obtain d' (which is both syntactically and semantically equivalent to d'_{NF}).

Since the rules applied to convert to normal forms are equivalences, we see that if it can be shown that $d_{NF} \vdash d'_{NF}$, then we have established

$$d \vdash d_{NF} \vdash d'_{NF} \vdash d'.$$

We focus on the part of the completeness proof that establishes $d_{NF} \vdash d'_{NF}$. Since d_{NF} is in normal form, by definition this means that

$$d_{NF} = \bigvee_{1 \leq i \leq n} d_i,$$

where each d_i is a unitary spider diagram containing only spiders that are single nodes. Similarly,

$$d'_{NF} = \bigvee_{1 \leq i \leq m} d'_i.$$

Returning to our consideration of regions, these normal forms ensure that, for each d_i and d'_j in d_{NF} and d'_{NF} respectively, the sets of regions are the same. That is, if we consider the underlying Euler diagrams, $L_i = L'_j$ and $R_i = R'_j$. So, the 'Euler part' of d_i is maximal with regard to the 'Euler part' of d_j and we have the right 'Euler conditions' for semantic entailment (i.e. if there were no spiders or shading then $d_i \vdash d'_j$). What remains is to consider the effects of spiders and shading. By comparing d_{NF} and d'_{NF}, an inference rule can be applied to d_{NF} in order to add spiders and shading to its components, increasing the number of diagrams in the disjunction, until it can be established that each unitary diagram, d_i, in the axiom logically entails a unitary diagram, d'_j, in the theorem. In this sense, the completeness proof strategy for spider diagrams maximizes the syntax in the axiom by adding curves (to get the 'right' curve label set), adding regions, and finally adding spiders and shading. Similar to the Euler diagram case, once this maximal form is achieved it is merely a matter of erasing syntax from d_i to obtain d'_j. We have $d_i \vdash d'_{NF}$ (by using an inference rule analogous to $P \vdash P \vee Q$ in propositional logic). Subsequently, it can be trivially shown that $d_{NF} \vdash d'_{NF}$, as required. We refer to [4] for full details.

For our purposes, it is sufficient for us to now define unitary spider diagrams where spiders comprise only single nodes, and to extend the definition of maximal to this case.

Definition 3. *A **spider diagram** is a tuple, $d = (L, R, R^*, S, \eta)$, where (L, R) is an Euler diagram, $R^* \subseteq R$ is a set of shaded regions, S is a finite set whose elements are called spiders and $\eta\colon S \to R$ is a function that identifies the region in which each spider is placed.*

In figure 3, the spider diagram d_2 has d of figure 2 as its underlying Euler diagram for which we previously specified L and R. In addition, there is one shaded region, and we have $R^* = \{(\{A, B\}, \{C, D\})\}$, two spiders, so $S = \{s_1, s_2\}$, and these spiders are placed in regions as given by $\eta(s_1) = (\{A, B\}, \{C, D\})$ and $\eta(s_2) = (\{C, D\}, \{A, B\})$. For diagrams with spiders comprising single nodes, the definition of maximal is as follows:

Definition 4. *Let $d = (L, R, R^*, S, \eta)$ and $d' = (L', R', R^{*\prime}, S', \eta')$ be spider diagrams such that $L = L'$. The diagram d is **maximal** with respect to d' provided $R' = R$, $R^{*\prime} \subseteq R^*$ and there exists an injection, $f\colon S' \to S$ such that for each $s' \in S'$, $\eta'(s') = \eta(f(s'))$.*

The definition of maximal given for spider diagrams generalizes that for Euler diagrams[2]. Intuitively, our definition of maximal is saying that *everything that occurs in d' also occurs in d*. The next lemma follows from a similar result in [4] (essentially restated here using our terminology):

Lemma 1. *Let $d = (L, R, R^*, S, \eta)$ and $d' = (L', R', R^{*\prime}, S', \eta')$ be spider diagrams such that $L = L'$ and $R = R'$. Suppose d is maximal with respect to d'. Then $d \vDash d'$ if and only if for each shaded region, r', in $R^{*\prime}$, the number of spiders in r' in both diagrams is the same.*

Theorem 1. *Let $d = (L, R, R^*, S, \eta)$ and $d' = (L', R', R^{*\prime}, S', \eta')$ be spider diagrams such that $L = L'$ and $R = R'$. If d is maximal with respect to d' and $d \vDash d'$ then $d \vdash d'$.*

Proof (Sketch). By lemma 1, each region that is shaded in d' contains the same number of spiders in d. Thus we can erase shading from d until $R^* = R^{*\prime}$, obtaining d_i, then remove spiders from d_i, that are not mapped to by the injective function $f\colon S(d') \to S(d)$. Finally, rename the spiders to obtain d'.

Theorem 2. *Let $d = (L, R, R^*, S, \eta)$ and $d' = (L', R', R^{*\prime}, S', \eta')$ be spider diagrams such that $L = L'$ and $R = R'$. If $d \vDash d'$ then d is maximal with respect to d'.*

Proof (Sketch). The proof is by contradiction. Suppose $d \vDash d'$ but d is not maximal with respect to d'. Then either $R^{*\prime} \not\subseteq R^*$ or there is no suitable injection, f, from the spiders of d' to those of d. In the first case, d' contains a shaded region, r, which is non-shaded in d. Then d' asserts that the set represented by r contains exactly n elements, where n is the number of spiders in r in d'. However, d allows the set represented by r to contain $n+1$ elements, so $d \nvDash d'$. In the second case, where there is no suitable injection, f, there is a region, r', in d' that contains more spiders than in d. Here, d asserts that the set represented by r' contains at least n elements, where n is the number of spiders in r' in d. However, d' asserts that the set represented by r' contains at least $n+j$ elements, where j is the number of 'extra' spiders in r' in d'. Again, $d \nvDash d'$. Thus, since $d \vDash d'$, the conditions for maximality must be satisfied.

We now have a completeness result concerning the fragment of the spider diagram logic that we have defined:

Theorem 3 (Completeness). *Let $d = (L, R, R^*, S, \eta)$ and $d' = (L', R', R^{*\prime}, S', \eta')$ be spider diagrams such that $L = L'$ and $R = R'$. If $d \vDash d'$ then $d \vdash d'$.*

Proof. If $d \vDash d'$ then, by theorem 2, d is maximal with respect to d'. By theorem 1, $d \vdash d'$.

To summarize, whilst the overall strategy is more complex, we still have a process of adding syntax to the axiom in order to maximize it with respect to the theorem we are aiming to prove.

[2] Note that for Euler diagrams we stipulated $R' \subseteq R$, but for spider diagrams we have $R = R'$. This difference is not significant; it merely makes the details of our argument more straightforward.

2.3 Constraint Diagrams

Constraint diagrams build on spider diagrams by adding further syntax, in particular arrows, to place constraints on binary relations. The system developed in [11] was shown to be sound and complete, with the completeness proof strategy directly extending that for spider diagrams. We omit formal definitions of these diagrams and maximal forms, and just illustrate the concepts by example.

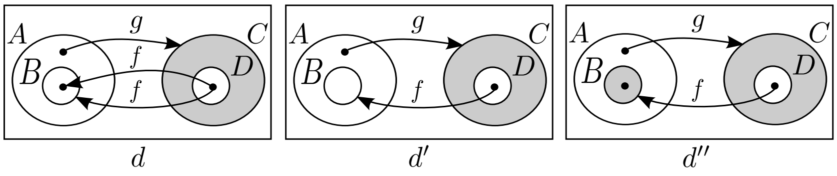


Fig. 4. Constraint diagrams: maximal forms and completeness.

First, to give a brief introduction to the meaning of arrows, consider d in figure 4. The arrow labelled g tells us that (the element represented by) the spider at its source is related to precisely the elements in C, the target, under (the relation represented by) g. Similarly, the spider in D is related to the unique element in B under f, and no other elements. In this example, d is maximal with respect to both d' and d''. Here, $d \vDash d'$ but $d \nvDash d''$. In the case of d and d', we can injectively map the spiders from d' to d in such a manner that the regions in which they are placed match. That is, there is an injective function $f\colon S' \to S$ where $\eta(s') = \eta(f(s'))$ and f ensures that the induced function $g\colon A' \to A$ is also an injection, where A' and A are the sets of arrows for d' and d respectively. Arrows are of the form $(\mathit{label}, \mathit{source}, \mathit{target})$ and $g(l, s, t) = (l, f(s), f(t))$ in the case where s and t are both spiders. We then delete shading, along with spiders and arrows that are not mapped to by f and g, from d to obtain d'.

Similar functions exist for d and d'', but this time we cannot apply inference rules to erase syntax from d to give d''. We would need to erase a spider from a shaded region, which is not sound; as with spider diagrams, the numbers of spiders in the shaded regions of the theorem, d', must match those in the axiom, d. Similarly to the spider diagram case, the number of spiders in the shaded regions of the theorem must be the same as in the axiom. These examples give the idea of the maximal forms used in constraint diagrams. We refer the reader to [11] for the full details which are too complex to illustrate in full here.

3 Concept Diagrams: The End of the Strategy?

We now consider extending the proof strategy described in the previous section to concept diagrams, which are intended to be used to model ontologies; see [5] for a practical example. They were introduced by Oliver et al. in 2009 [8] and extend constraint diagrams. Concept diagrams may include unlabelled curves, which we call *anonymous curves* and which represent anonymous subsets of the universal set. These provide an increase in expressiveness over notations such as constraint diagrams. As with our consideration of spider diagrams, we only permit spiders to comprise single nodes. Thus, taking a concept diagram from this fragment and removing its arrows and anonymous curves yields a spider diagram from the fragment defined in section 2.2.

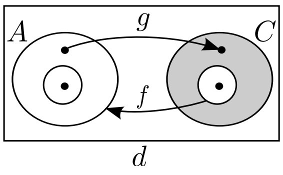


Fig. 5. A concept diagram.

The diagram in figure 5 is a concept diagram. The part of the diagram made up of labelled curves, shading and spiders is a spider diagram. The arrows provide information about binary relations. The diagram d expresses the following, in addition to the information given in the underlying spider diagram:

1. there are two sets, x and y, the former is a subset of A and the latter is a subset of C,
2. the image of the relation f, when its domain is restricted to y, is A,
3. there is an element, a, in $A - x$ such that the image of the relation g, when its domain is restricted to a, is the element in $C - y$.

We now present the syntax of the fragment of concept diagrams under consideration, adapted from [10].

Definition 5. *A* ***unitary concept diagram*** *is a tuple* $d = (L, C, R, R^*, S, \eta, A)$, *where*

1. $L = L(d)$ *is a finite set whose elements are called labelled curves,*
2. $C = C(d)$ *is a finite set whose elements are called anonymous curves,*
3. $R = R(d)$ *is a set of regions such that*
$$R \subseteq \{(in, (L \cup C) - in) : in \subseteq L \cup C\}.$$
4. $R^* = R^*(d) \subseteq R$ *is a set of shaded regions.*
5. $S = S(d)$ *is a finite set whose elements are called spiders,*
6. $\eta = \eta_d \colon S \to R$ *is a function that returns the location of each spider.*
7. $A = A(d)$ *is a finite set of arrows, each of the form* (l, s, t), *where* l *is the label,* $s \in L \cup C$ *is the source and* $t \in S \cup L \cup C$ *is the target.*

If $d = (L, C, R, R^*, S, \eta, A)$ is a concept diagram and $C = \emptyset$ then $d_s = (L, R, R^*, S, \eta)$ is a spider diagram. Semantics are assigned to concept diagrams similarly to the previous notations discussed: in addition to the usual interpretation of the underlying spider diagram, the arrows of a concept diagram place restrictions on binary relations and the anonymous curves represent the existence of sets as illustrated in our examples[3].

In order to extend the strategy discussed in the previous section to concept diagrams, we first extend the definition of maximality:

Definition 6. *Let $d = (L, C, R, R^*, S, \eta, A)$ and $d' = (L', C', R', R^{*\prime}, S', \eta', A)$ be concept diagrams such that $L = L'$. The diagram d is **maximal** with respect to d' provided:*

1. *there exists a bijection $g\colon L' \cup C' \to L \cup C$ such that*
   (a) *g is the identify map when its domain is restricted to L',*
   (b) *g induces a bijection $h\colon R' \to R$, defined by $h(in, out) = (in', out')$, where*
      i. *$in' = \{g(c') : c \in in\}$, and*
      ii. *$out' = \{g(c') : c' \in out\}$,*
      which ensures for each $(in, out) \in R^{*\prime}$, $h(in, out) \in R^*$,
2. *there exists an injection, $f\colon S' \to S$ such that for each $s' \in S'$, $\eta'(s') = \eta(f(s'))$, and*
3. *g and f induce an injection $p\colon A' \to A$ defined by*

$$p(l, s, t) = \begin{cases} (l, g(s), g(t)) & \text{if } s, t \in L' \cup C' \\ (l, g(s), f(t)) & \text{if } s \in L' \cup C' \wedge t \in S' \\ (l, f(s), g(t)) & \text{if } s \in S' \wedge t \in L' \cup C' \\ (l, f(s), f(t)) & \text{if } s, t \in S'. \end{cases}$$

Equipped with the definition of maximality, we can examine how to generalize the lemma and theorems from section 2.2 in order to extend the strategy. We start by considering the equivalent of lemma 1:

Conjecture 1. Let $d = (L, C, R, R^*, S, \eta, A)$ and $d' = (L', C', R', R^{*\prime}, S', \eta', A')$ be concept diagrams such that $L = L'$. Suppose d is maximal with respect to d'. Then $d \vDash d'$ if and only if for each shaded region, r', in $R^{*\prime}$, the number of spiders in r' is the same as the number of spiders in $h(r')$ in d.

Figure 6 shows a counterexample to conjecture 1. First, it is obvious that d_1 is maximal with respect to d_2. Here, d_1 tells us that there exists a set containing exactly two elements. From this we can deduce that there exists a set containing exactly one element. That is, d_2 follows logically from d_1. The shaded region in d_2 contains *fewer* spiders than in d_1. In both these diagrams, we see that the shading is actually redundant: removing the shading does not alter the informational content of the diagrams.

[3] Here, we note that our representation of concept diagrams assumes that spiders represent the existence of elements; strictly, in concept diagrams, spiders act as free variables. Formally, unitary diagrams as we have defined them would need to be prefixed by existential quantifies (one for each spider) to get our interpretation. However, to avoid diagram clutter, we simply omit the existential quantifiers since no ambiguity arises.

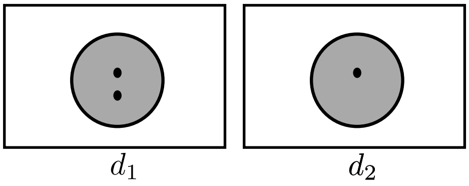


Fig. 6. The spiders in the shaded regions do not match.

Clearly, such problems concerning spiders and shading impact our ability to obtain a completeness result for the fragment under consideration. In order to obtain completeness, we need inference rules that allow us to identify when (a) shading is redundant, (b) we can delete spiders from shaded regions, and (c) when anonymous curves are redundant. Worthy of note is that the diagrams in figure 6 are semantically equivalent to spider diagrams (on removing the anonymous curves, the informational content is unaltered). Thus, the problems here arise from the syntactic richness of the notation and are not merely because of an increase in expressive power.

To proceed with our exposition of problems that arise when attempting to extend the previously used proof strategies to concept diagrams, we extend the definition of maximal to incorporate the condition on shading given in conjecture 1:

Definition 7. *A concept diagram d is* ***strongly maximal*** *with respect to d' provided*

1. d is maximal with respect to d', and
2. for each shaded region, r', in $R^{*\prime}$, the number of spiders r' is the same as the number of spiders in $h(r')$ in d.

By doing this, we are following the standard mathematical process of applying further constraints to a conjecture for which we have found a counterexample. In fact, conjecture 1 is trivially true in the strongly maximal case. As a consequence, our attempts to extend the completeness strategy apply to a smaller fragment of concept diagrams.

Next, we consider extending theorem 1 to concept diagrams:

Conjecture 2. Let $d = (L, C, R, R^*, S, \eta, A)$ and $d' = (L', C', R', R^{*\prime}, S', \eta', A')$ be concept diagrams such that $L = L'$. If d is strongly maximal with respect to d' and $d \vDash d'$ then $d \vdash d'$.

To establish the truth of conjecture 2, we start by defining the inference rules which are needed to establish $d \vdash d'$, if d is strongly maximal with respect to d' and $d \vDash d'$.

Inference rule 1: Remove arrow. Let $d = (L, C, R, R^*, S, \eta, A)$ be a concept diagram and let $a \in A$ be an arrow in d. Let $d' = (L, C, R, R^*, S, \eta, A - \{a\})$ be the diagram obtained by removing a from d. Then d logically entails d'.

Inference rule 2: Remove shading. Let $d = (L, C, R, R^*, S, \eta, A)$ be a concept diagram and let $r^* \in R^*$ be a shaded region in d. Let $d' = (L, C, R, R^* - \{r^*\}, S, \eta, A)$ be the diagram obtained by removing the shading from r^* in d. Then d logically entails d'.

Inference rule 3: Remove spider. Let $d = (L, C, R, R^*, S, \eta, A)$ be a concept diagram and let $x \in S$ be a spider with an non-shaded location in d. Let $d' = (L, C, R, R^*, S - \{x\}, \eta, A)$ be the diagram obtained by removing x from d. Then d logically entails d'.

Inference rule 4: Substitute spider Let $d = (L, C, R, R^*, S, \eta, A)$ be a concept diagram and let $x \in S$. Let y be a spider not in S. Let $d' = (L, C, R, R^*, (S - \{x\}) \cup \{y\}, (\eta - \{(x, \eta(x)\}) \cup \{(y, \eta(x)\}, A)$ be the diagram obtained by replacing x with y in d_1. Then d is logically equivalent d'.

Inference rule 5: Substitute anonymous curve Let $d = (L, C, R, R^*, S, \eta, A)$ be a concept diagram and let $c \in C$. Let c' be an anonymous curve not in C. Let d' be the diagram obtained from d by replacing all occurrences of c with c'. Then d is logically equivalent d'.

Lemma 2. *The inference rules are sound.*

We have sufficient inference rules to show that conjecture 2 is true.

Theorem 4. *Let $d = (L, C, R, R^*, S, \eta, A)$ and $d' = (L', C', R', R^{*\prime}, S', \eta', A')$ be concept diagrams such that $L = L'$. If d is strongly maximal with respect to d' and $d \vDash d'$ then $d \vdash d'$.*

Proof (Sketch). Assume d is strongly maximal with respect to d'. By the definition of strong maximality there is an injection, p, from the arrows of d' to the arrows of d. Therefore, we can apply rule 1, remove arrow, to d until its arrows match those of d', i.e. p becomes bijective. Similarly, we can apply rule 2, remove shading, repeatedly until the shading of d matches that of d'. Now, by the definition of strong maximality, each shaded region, r', in d', contains the same number of spiders as $h(r')$ in d. This means we can apply rule 3 to remove spiders until f is bijective. All that differs now are the spiders and the anonymous curves. Apply rules 4 and 5 to obtain d'.

We must now consider whether theorem 2 extends to concept diagrams:

Conjecture 3. Let $d = (L, C, R, R^*, S, \eta, A)$ and $d' = (L', C', R', R^{*\prime}, S', \eta', A')$ be concept diagrams such that $L = L'$. If $d \vDash d'$ then d is strongly maximal with respect to d'.

Figure 6 provides a counterexample to conjecture 3 (as well as conjecture 1). The problems arising from counterexamples like figure 6 may be easy to overcome (by defining inference rules that remove redundant anonymous curves, for instance). We will now demonstrate that problems also arise in more complex situations where arrows are involved.

Such a counterexample to conjecture 3 can be seen in figure 7. In d, the anonymous curves x and y are given labels for convenience. The arrow (f, A, B) tells us the image of f when its domain is restricted to A is B. One of the elements inside x is related to nothing under f, which we know by the arrow targeting the curve that represents the empty set (i.e. the curve containing shading but no spiders). At least one of the other elements inside A must therefore be related to the element inside B. In d', the arrows provide this information that we have just deduced from the arrows of d. The other information provided by d' 'agrees' with that provided by d, so $d \vDash d'$. However, d is not strongly maximal with respect to d': there is no appropriate injective mapping from arrows of d' to those of d.

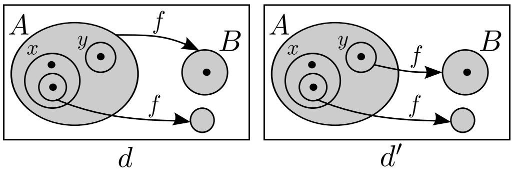


Fig. 7. Conjecture 3: not enough arrows.

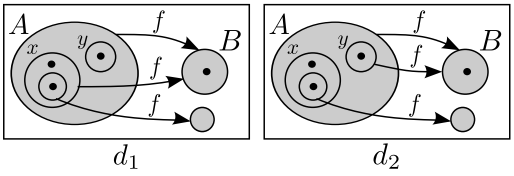


Fig. 8. Adding arrows.

As stated above, with regard to shading and spiders we can attempt to overcome the problems by devising inference rules for removing redundant anonymous curves, for example. With regard to arrows, the question arises as to whether we can add arrows to diagram d, figure 7, until there is an appropriate injection from the arrows of d' to those of d. This leads to the notion of *potential arrows*, those arrows which can be added to a diagram without changing its meaning:

Definition 8. *Let* $d = (L, C, R, R^*, S, \eta, A)$ *be a concept diagram and let* $a \notin A$ *be an arrow not in* d. *Let* $d' = (L, C, R, R^*, S, \eta, A \cup \{a\})$. *If* d *is semantically equivalent to* d' *then* a *is a* ***potential arrow*** *for* d.

In figure 7, there are two potential arrows for d. The arrow (f, A, B) tells us that at least one element of A is related under f to the element in B, and so we can add an arrow which represents this information explicitly. The arrows (f, x, B) in diagram d_1 and (f, y, B) in diagram d_2, figure 8, are potential arrows for d, since either arrow can be added to d without changing its meaning. After adding either arrow, however, the other arrow ceases to be a potential arrow. Neither d_1 nor d_2 have any potential arrows. Thus, an element of choice arises when adding potential arrows to concept diagrams, which again causes problems for completeness. Rather, adding potential arrows results in a set of obtainable diagrams each of which is semantically equivalent to the original diagram. In figure 8, $\{d_1, d_2\}$ is the set of such diagrams obtainable from d (figure 7).

If we are to extend the completeness proof strategy by adding arrows to the axiom, *all* of the diagrams obtained from d using this process must have an arrow set that can be injectively mapped to by the arrows of d' in the appropriate way; this is because the diagrams obtained are semantically equivalent to d and, therefore, semantically entail d'. We can see that we can remove syntax from d_2 to obtain d' since d_2 is strongly maximal with respect to d'. However , there is currently no sequence of rules that would, or general strategy that can be used to, transform d_1 into d', even though $d_1 \vDash d'$ (since d_1 is semantically equivalent to d and $d \vDash d'$); d_1 is not strongly maximal with respect to d'.

A possible approach to overcome this problem is to determine whether we can remove syntax from d' without changing its meaning until we have an appropriate injection from its arrows to those of, in this example, d_1. Unfortunately, no arrows can be removed from d' without weakening information, so such an approach is still insufficient.

Compared to the steps required to extend the definition of maximality, the non-uniqueness of the ways in which we can add arrows is the most serious blow so far to the aim of extending the completeness proof strategy. It is not at all clear how we need to change the syntax of an arbitrary axiom, d, to obtain d' in general. As we have demonstrated, we need to devise strategies for altering the spiders, shading and the arrows present in either the axiom and/or theorem until the axiom is strongly maximal with respect to the theorem. Even once this is solved, it will be challenging to extend the completeness proof strategies to larger fragments of the concept diagram logic.

4 Conclusion

We have identified commonality in the completeness proof strategies of various logics based on Euler diagrams and shown how, as expressiveness increases, the strategy readily extends in some cases. We have illustrated various ways in which this strategy breaks down for concept diagrams, which are syntactically richer and more expressive than earlier logics based on Euler diagrams. The problems identified with extending the completeness strategy to concept diagrams arise because we cannot simply delete syntax from the axiom to obtain the theorem, even for the very small fragment that we considered. Thus, we have established that the existing completeness proof strategies are limited. The non-unique ways of adding syntax to concept diagrams, which further complicates the issue, results from the syntactic richness of the notation and from their expressive power. We believe that the same phenomena will arise in equally expressive logics.

We examined ways in which parts of the completeness proof strategy might be ‘patched up’ but we conjecture that a new strategy needs to be developed. One (rather undesirable) route to obtaining completeness for fragments of concept diagrams is to derive inference rules for them inspired by complete symbolic logics[4]. This is not the route we want to pursue, strongly preferring a set of inference rules that makes use of *diagrammatic* reasoning. Even small fragments of the more expressive visual logics will require new completeness strategies.

We believe the need for expressive visual logics such as concept diagrams is clear, since they allow the techniques of diagrammatic reasoning to be applied in new domains, such as ontology specification. For these logics to be fully exploited, we need to develop sound inference rules with clearly understood metatheories, including establishing expressiveness and identifying complete fragments. Understanding the effect that increases in both syntactic richness and notational expressiveness have on completeness is essential for the informed design of new logics.

References

1. A. Fish, J. Flower, and J. Howse. The semantics of augmented constraint diagrams. *Journal of Visual Languages and Computing*, 16:541–573, 2005.
2. J. Gil, J. Howse, and S. Kent. Formalising spider diagrams. In *IEEE Symposium on Visual Languages*, pages 130–137. IEEE, 1999.
3. E. Hammer. *Logic and Visual Information.* CSLI Publications, 1995.
4. J. Howse, G. Stapleton, and J. Taylor. Spider diagrams. *LMS Journal of Computation and Mathematics*, 8:145–194, 2005.
5. J. Howse, G. Stapleton, K. Taylor, and P. Chapman. Visualizing ontologies: A case study. In *International Semantic Web Conference 2011*, pages 257–272. Springer, 2011.
6. S. Kent. Constraint diagrams: Visualizing invariants in object oriented modelling. In *Proceedings of OOPSLA97*, pages 327–341. ACM Press, October 1997.
7. K. Mineshima, M. Okada, Y. Sato, and R. Taakemura. Diagrammatic reasoning system with euler circles: Theory and experiment design. In *Diagrams*, pages 188–205. Springer, 2008.
8. I. Oliver, J. Howse, G. Stapleton, E. Nuutila, and S. Torma. Visualising and specifying ontologies using diagrammatic logics. In *5th Australasian Ontologies Workshop*, volume 112, pages 87–104. CRPIT, 2009.
9. S.-J. Shin. *The Logical Status of Diagrams.* Cambridge University Press, 1994.
10. G. Stapleton, J. Howse, P. Chapman, I. Oliver, and A. Delaney. What can concept diagrams say? In *Submitted to Diagrams 2012*, 2012.
11. G. Stapleton, J. Howse, and J. Taylor. A decidable constraint diagram reasoning system. *Journal of Logic and Computation*, 15(6):975–1008, December 2005.
12. 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.
13. 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.

[4] Symbolic logics have radically different (uncomparable) completeness proof strategies due to their vastly different syntax.

On the Cognitive Efficacy of Euler Diagrams in Syllogistic Reasoning: A Relational Perspective

Koji Mineshima[1], Yuri Sato[1], Ryo Takemura[2], and Mitsuhiro Okada[1]

[1] Department of Philosophy, Keio University
2-15-45 Mita, Minato-ku, Tokyo 108-8345, Japan.
{minesima,sato,mitsu }@abelard.flet.keio.ac.jp
[2] College of Commerce, Nihon University
5-2-1 Kinuta, Setagaya-ku, Tokyo 157-8570, Japan.
takemura.ryo@nihon-u.ac.jp

Abstract. Although logic diagrams are widely used as methods for introducing students to elementary logical reasoning, it is still open to debate in cognitive psychology whether diagrams can aid untrained people to successfully conduct deductive reasoning. In our previous work, some empirical evidence was provided for the effectiveness of a certain type of logic diagrams in the process of solving categorical syllogisms. However, the question of why certain diagrams but not others have such inferential efficacy in performing syllogism reasoning has not been fully answered. Based on a proof-theoretical analysis of categorical syllogisms and diagrammatic reasoning, we supplement our previous study of cognitive efficacy of diagrams and argue that the relational information underlying quantified sentences plays a crucial role in understanding the efficacy of diagrams in syllogistic reasoning. The distinctive features of our conception of diagrammatic reasoning are made clear by comparing it with the model-theoretic conception of ordinary reasoning developed in the mental model theory.

1 Introduction

In logic teaching, Venn and Euler diagrams have been widely used as tools for introducing students to elementary logical reasoning, including set-theoretical and syllogistic reasoning.[3] However, in the literature of cognitive psychology of reasoning, it is still open to debate whether external diagrams can aid logically untrained people to conduct deductive reasoning in a successful way (see Scaife & Rogers [30] for an overview of the work on external representations in cognitive science). Indeed, it is often claimed that diagrams can only serve as an auxiliary source of information in deductive problem solving. Thus, Larkin and Simon [14], in a seminal work on the efficacy of diagrammatic representations in problem solving in general, argued that reasoning is largely independent of ways of representing information, and hence, that diagrams are less beneficial in reasoning than in such tasks as searching and recognition. Additionally, previous studies reported empirical evidence for negative effects of traditional Euler diagrams on the performance of syllogistic reasoning (Calvillo, Deleeuw, & Revlin [4];

[3] In fact, Leonhard Euler [7] introduced his diagrams to teach Aristotelian syllogistic logic to a German princess.

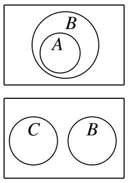
3rd International Workshop on Euler Diagrams, July 2, 2012, Canterbury, UK.

Rizzo & Palmonari [25]). Furthermore, although various systems of logic diagrams have been proposed and studied using the methods of mathematical logic (e.g. Shin [31]; Hammer [11]; for a survey, see Stapleton [34]), little attention has been paid to the question of how effective such diagrammatic systems are in people's actual reasoning.[4]

To improve this situation, we have studied how logic diagrams can support actual deductive reasoning, focusing on the case of syllogistic reasoning supported by Euler and Venn diagrams that are externally given to reasoners (Sato, Mineshima & Takemura [26, 27]).[5] Typical examples of reasoning tasks that we examined are shown in Figs.1 and 2.

All *A* are *B*.

No *C* are *B*.

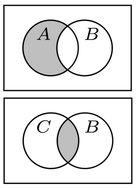


(Therefore, No *C* are *A*)

Fig.1 An example of a syllogistic reasoning task with Euler diagrams

All *A* are *B*.

No *C* are *B*.



(Therefore, No *C* are *A*)

Fig.2 An example of a syllogistic reasoning task with Venn diagrams

Euler diagrams represent set relationships in terms of inclusion and exclusion relations between circles (see the diagrams in Fig.1). By contrast, Venn diagrams have a fixed configuration of circles and represent set relationships by stipulating that shaded regions denote the empty set (see the diagrams in Fig.2). In the experiments of Sato et al. [26], subjects were divided into three groups, called the Euler group, Venn group, and Linguistic group. The Euler group and Venn group were first provided with instructions on the meanings of diagrams. A pretest was conducted to check whether the subjects understood the instructions correctly. The Euler group was then asked to solve syllogistic reasoning tasks in which subjects were presented with two sentential premises together with two corresponding Euler diagrams, as in Fig. 1, and asked to choose a valid conclusion. Similarly, the Venn group was asked to solve tasks like the one in Fig. 2. The Linguistic group was presented only with sentential premises and required to choose a valid conclusion without any aid from diagrams. The results showed that (1) the performance of the Euler and Venn groups was significantly better than that of the

[4] A notable exception is important work on *hyperproof* by Stenning, Cox, and Oberlander [36], where the effects of teaching elementary logic classes using Hyperproof methods, i.e., multimodal graphical and sentential methods, and a standard syntactic teaching method are compared.

[5] Although traditional syllogisms are less expressive than standard first-order logic, they are one of the most basic form of natural language inferences and still important for investigating human reasoning. Indeed, syllogistic logics, considered as alternative logical systems to standard first-order logic, have recently attracted increasing attention from logical and linguistic points of view, for example, in the study of decidable fragments of first-order logic; see Moss [21] and references given there.

Linguistic group, and (2) the performance of the Euler group was significantly better than that of the Venn group.

Sato et al. [26, 27] argue that the differences in performance between the three groups can be explained on the basis of the distinction between two kinds of efficacy, namely, interpretational and inferential efficacy. By interpretational efficacy we mean the effects of diagrams on determining the correct interpretation of a sentence. For example, sentence *All B are A* tends to be interpreted as equivalent to *All A are B*.[6] Diagrams can contribute to avoiding deductive reasoning errors due to such unintended interpretations of linguistic materials. For example, those who are presented with diagrams representing *All B are A* as shown in Figs. 1 and 2 can immediately see that the semantic information delivered is not equivalent to *All A are B* by virtue of their form. Diagrams can also can play a crucial role in reasoning processes. We refer to the efficacy of diagrams in reasoning processes themselves as "inferential efficacy". More specifically, when diagrams of a certain form are externally given, the process of solving deductive reasoning tasks could be replaced with the syntactic manipulation of diagrams.

It should be noted that in the experimental set-up of Sato et al. [26], subjects in the Euler and Venn groups were given instructions on the meaning of diagrams, while subjects in the Linguistic group were not. Then one might argue that the difference in training could have had a major effect on differences in performance between the Euler and Venn groups, on the one hand, and the Linguistic group, on the other. However, such an objection can be avoided if a comparison is made between the Euler group and the Venn group. The latter was also given substantial instructions and practice trials, yet the result showed that the performance of the Euler group was significantly better than that of the Venn group.

The hypothesis explored in Sato et al. [26] was that the difference in performance between the three groups can be explained by assuming that Venn diagrams only have interpretational efficacy, while Euler diagrams have both interpretational and inferential efficacy.[7] That is, Euler diagrams not only contribute to the correct interpretations of categorical sentences but also play a substantial role in the inferential processes of solving syllogisms. To substantiate this claim, Sato et al. [27] outlined a cognitive model of syllogistic inferences that are externally supported by diagrams, assuming that both (categorical) sentences and diagrams conventionally express semantic information, and furthermore, that diagrams are syntactic objects to be manipulated in reasoning processes. In Sato et al. [27], however, the semantic and syntactic (proof-theoretical) analyses of categorical sentences and diagrams were left unspecified. Consequently, the question of why certain logic diagrams, in particular, Euler diagrams, have inferential efficacy in syllogism solving has not been fully answered. The rest of the present paper is devoted to addressing this question. Building on the proof-theoretical study of categorical syllogisms and Euler diagrams in Mineshima, Okada and Takemura [19, 20], we

[6] This is known as "illicit conversion error" in the literature; see, e.g. Newstead & Griggs [22].

[7] Gurr [10] emphasizes that in addition to the process of combining information, the process of extracting information from a diagram plays a role in diagrammatic reasoning. Sato, Mineshima, and Takemura [28] examines differences in the cognitive process of extracting information between Euler and Venn diagrams in some details.

will analyze both syllogistic and diagrammatic inferences from a unified perspective, which we call a *relational* perspective. Thus, the aim of this paper is to make a connection between the logical study of syllogisms and diagrams in Mineshima et al. [19, 20] and the cognitive study of diagrams in Sato et al. [26, 27], and thereby to provide a model of reasoning in which the experimental results of Sato et al. [26, 27] can be explained in a natural way. The key assumption is that both syllogistic and diagrammatic inferences are decomposed as inferences with two primitive relations, i.e., *inclusion* and *exclusion*. We claim that the efficacy of Euler diagrams in syllogistic reasoning derives from the fact that they are effective ways of representing and reasoning about relational structures that are implicit in categorical (quantified) sentences.

The formal study of logic diagrams in Mineshima, Okada and Takemura [20] also sheds light on the question of how diagrams can contribute to judging that a given inference is *invalid* in actual reasoning. It has been noticed in cognitive psychology of reasoning that falsification tasks, including tasks that require a reasoner to judge that there is no valid conclusion drawable from a given set of premises, are often difficult for untrained people when inference materials are only presented in linguistic (sentential) form. Interestingly, the experimental results in Sato et al. [26] showed that Euler diagrams were particularly effective in supporting such falsification tasks of syllogistic reasoning. We argue that the efficacy of Euler diagrams in falsification tasks is partly explained by assuming that when such diagrams are externally given, the information that there is no valid conclusion drawable from the premise diagrams can be obtained in a direct way, specifically, by combining premise diagrams and extracting the relevant relational information. This way of understanding diagrammatic reasoning can be made clear by comparing it with model-based inferences such as those studied in the mental model theory (e.g., Johnson-Laird & Byrne [13]), where the process of constructing a particular model plays a crucial role in checking the validity and invalidity of an inference. By taking a closer look at the difference between the two conceptions of inferences, we will point out that in reasoning with Euler diagrams, constraints on unification processes of diagrams play an important role; furthermore, both processes of proving and refuting a conclusion can be realized as a uniform process of syntactic manipulation of diagrams.

This paper is structured as follows. In Section 2, we provide a preliminary background on a relational analysis of categorical syllogisms, originally provided in Mineshima, Okada and Takemura [20]. In Section 3, we turn to the relational analysis of Euler diagrams. In Section 4, our model of diagrammatic reasoning is compared with that of the mental model theory. Finally, in Section 5, we give a summary of the discussion.

2 Background: Categorical syllogisms as relational inferences

Categorical syllogisms are inferences concerned with *quantificational* sentences in natural languages. According to the traditional analysis in logic textbooks, such quantificational sentences are analyzed as formulas in first-order logic, i.e., formulas involving quantification over individuals. Thus, *All A are B* is analyzed as $\forall x(Ax \to Bx)$ and *Some A is B* as $\exists x(Ax \wedge Bx)$, and so on. By contrast, according to the theory of *generalized* *quantifiers* (see Barwise & Cooper [2]), which is dominant in the field of natural language semantics, quantificational expressions such as *every*, *some*, and *no* are analyzed as denoting *relations* between sets. Thus, a universal sentence of the form *All A are B* is semantically analyzed as expressing that $\mathbf{A} \subseteq \mathbf{B}$, where the determiner *all* corresponds to the subset relation. Similarly, *No A are B* is analyzed as expressing that $\mathbf{A} \cap \mathbf{B} = \emptyset$, where the determiner *no* corresponds to the disjointness relation.[8] Proof systems for such a relational semantics of quantificational sentences have been investigated in the modern reconstructions of Aristotelian syllogisms (cf. Łukasiewicz [16]; Corcoran [6]; Smiley [33]) and in the recent development of natural logic (cf. Moss [21]). In these studies, the relational structure of a quantified sentence is taken as a primitive logical form; as a result, syllogistic inferences are formalized as a certain kind of *relational* inference without reference to first-order quantifiers and individual terms.

Mineshima, Okada, and Takemura [20] present a simple proof system based on two primitive relations, i.e., inclusion $\sqsubset$ and exclusion $\vdash\!\dashv$ for syllogistic inferences.[9] The system is called a *generalized syllogistic inference system* and abbreviated as GS. The inference system of GS is simple but expressive enough to represent categorical syllogisms in a perspicuous way. In the rest of this section, we provide a brief overview of the syntax of GS and then see how to formalize categorical syllogisms using inclusion and exclusion relations of GS.

The language of GS is defined as follows. Terms of GS (denoted by $X, Y, Z, \dots$) are divided into singular terms (denoted by $a, b, c, \dots$) that correspond to proper names like Socrates, and general terms (denoted by $A, B, C, \dots$) that correspond to common nouns like philosopher. An atomic formula (denoted by $P, Q, \dots$) is of the form $X \sqsubset Y$ or $X \vdash\!\dashv Y$, where X and Y are terms. A complex formula (denoted by $\mathcal{P}, \mathcal{Q}, \dots$) is defined as a set of atomic formulas, $\{P_1, \dots, P_n\}$. Intuitively, $\{P_1, \dots, P_n\}$ means the conjunction of atomic formulas, $P_1 \wedge \cdots \wedge P_n$. To simplify the notation, we usually omit the brackets.

The proof system of GS is shown in Fig. 3. A proof in GS has a tree form; it starts with formulas of GS or Axioms (*ax*) and proceeds by one of the inference rules in Fig. 3.[10] As we will see below, the crucial rules for representing categorical syllogisms are the ($\sqsubset$) and ($\vdash\!\dashv$) rules. A set-theoretical semantics of GS can be given in a natural way, but to conserve space we omit it here. See Mineshima, Okada & Takemura [19, 20], where soundness and completeness are established.

Now we turn to categorical syllogisms. A categorial sentence has one of the following forms: *All A are B*, *No A are B*, *Some A are B*, and *Some A are not B*, where A and B are distinct general terms. We assume that readers are familiar with what count as valid inferences in categorical syllogisms.[11] A translation $(\cdot)^\circ$ from a categorical sentence

8 A strong argument against the traditional first-order analysis of natural language quantifiers comes from the fact that proportional quantifiers such as *most* and *half of* cannot be properly represented in first-order logic. See Barwise & Cooper [2].

9 The notation of $\vdash\!\dashv$ is due to Gergonne [9], where symbols for some binary relations (the so-called "Gergonne relations") were introduced for the purpose of the abstract representation of Euler diagrams.

10 The (C) rule allows us to infer $a \sqsubset A$ ("a is A") from $A \sqsubset a$ ("Only a is A") and $a \sqsubset b$ ("a is b") from $b \sqsubset a$ ("b is a").

11 See Mineshima et al. [20] for discussion on so-called existential import.

Axiom (ax): $X \sqsubset X$.
Inference rules:

$$\dfrac{X \sqsubset Y \quad Y \sqsubset Z}{X \sqsubset Z}\ (\sqsubset) \qquad \dfrac{X \sqsubset Y \quad Y \mathrel{\vdash\!\dashv} Z}{X \mathrel{\vdash\!\dashv} Z}\ (\mathrel{\vdash\!\dashv}) \qquad \dfrac{X \sqsubset a}{a \sqsubset X}\ (\mathsf{C})$$

$$\dfrac{\mathcal{P} \quad \mathcal{Q}}{\mathcal{P} \cup \mathcal{Q}}\ (+) \qquad \dfrac{\mathcal{P}}{\mathcal{Q}}\ (-)$$

where in $(+)$, $\mathcal{P} \neq \mathcal{Q}$, and in $(-)$, $\mathcal{Q}$ is a proper subset of $\mathcal{P}$.

Fig.3 Proof system of GS

into a GS-formula is defined as follows.

$$\begin{aligned}
(\mathit{All\ A\ are\ B})^\circ &= A \sqsubset B \\
(\mathit{No\ A\ are\ B})^\circ &= A \mathrel{\vdash\!\dashv} B \\
(\mathit{Some\ A\ are\ B})^\circ &= \{c \sqsubset A,\, c \sqsubset B\} \text{ for some } c \\
(\mathit{Some\ A\ are\ not\ B})^\circ &= \{d \sqsubset A,\, d \mathrel{\vdash\!\dashv} B\} \text{ for some } d
\end{aligned}$$

where c and d are arbitrarily chosen singular terms. The crucial point is that in GS existential sentences are *decomposed* in terms of inclusion and exclusion.[12] Given this translation, all the valid inferences in categorical syllogism can be transformed into the proofs in GS. Let us look at some typical examples. To begin with, syllogisms Barbara (*All A are B*, *All B are C*. Therefore, *No A are C*) and Celarent (*All A are B*, *No B are C*. Therefore, *No A are C*) correspond to the $(\sqsubset)$ and $(\mathrel{\vdash\!\dashv})$ rules, respectively.

$$\dfrac{\overset{\text{All } A \text{ are } B}{A \sqsubset B} \qquad \overset{\text{All } B \text{ are } C}{B \sqsubset C}}{\underset{\text{All } A \text{ are } C}{A \sqsubset C}}\ (\sqsubset) \qquad\qquad \dfrac{\overset{\text{All } A \text{ are } B}{A \sqsubset B} \qquad \overset{\text{No } B \text{ are } C}{B \mathrel{\vdash\!\dashv} C}}{\underset{\text{No } A \text{ are } C}{A \mathrel{\vdash\!\dashv} C}}\ (\mathrel{\vdash\!\dashv})$$

Here, to make clear the translation between categorical sentences and formulas of GS, we attach a categorical sentence with each assumption and conclusion. As a case involving an existential sentence, consider a syllogism Darii (*Some A are B*, *All B are C*. Therefore, *Some A are C*). This inference is simulated in GS as follows:

$$\dfrac{\dfrac{\overset{\text{Some } A \text{ are } B}{a \sqsubset A,\, a \sqsubset B}}{a \sqsubset A}\ (-) \qquad \dfrac{\dfrac{\overset{\text{Some } A \text{ are } B}{a \sqsubset A,\, a \sqsubset B}}{a \sqsubset B}\ (-) \qquad \overset{\text{All } B \text{ are } C}{B \sqsubset C}}{a \sqsubset C}\ (\sqsubset)}{\underset{\text{Some } A \text{ are } C}{a \sqsubset A,\, a \sqsubset C}}\ (+)$$

As stated above, the formula "$a \sqsubset A, a \sqsubset B$" means the *conjunction* of $a \sqsubset A$ and $a \sqsubset B$. Hence, the $(+)$ and $(-)$ rules can be understood as corresponding to introduction and

[12] This translation is similar to Aristotle's alternative way of formulating categorical syllogisms, known as *ecthesis*. See Łukasiewicz [16] for a modern reformulation of *ecthesis*.

elimination rules of conjunction in standard natural deduction systems (i.e., the rule which allows to infer $P \wedge Q$ from P and Q and the rule which allows to infer P as well as Q from $P \wedge Q$). By decomposing existential sentences in terms of inclusion and exclusion, we can represent syllogisms like Darii without using some additional rules specific to existential sentences; if we take existential sentences as primitive formulas or define them from other formulas using sentential negation (e.g. *some A are not B* is defined as *not (all A are B))*, we will need such additional axioms or inference rules. [13] It turns out that all the valid categorical syllogisms (with and without existential import) can be simulated in GS; more specifically, they can be proved using the inference rules $(\sqsubset)$, $(\vdash\!\dashv)$, $(+)$, and $(-)$ only.[14]

If the relational information encoded by categorical sentences was transparent to untrained reasoners, it would be much easier for them to solve categorical syllogisms. However, the cognitive psychological studies of deductive reasoning accumulated so far showed that this is not the case (see Sato et al. [26] and references given there). For example, the fact that logically untrained people often interpret *All A are B* as equivalent to *All B are A* indicates that the relational information $A \sqsubset B$ is not directly available to them. Similarly, the observed difficulties in solving categorical syllogisms involving existential sentences (cf. Evans, Newstead & Byrne [8]) suggest that there is a certain gap between ordinary ways of performing existential inferences and relationally decomposed processes as indicated above.

3 Solving categorical syllogisms using diagrams

As mentioned in Section 1, there are two aspects in which diagrams can externally support ordinary reasoning. Given the relational analysis of categorical syllogisms in the last section, we can summarize the effectiveness of Euler diagrams in syllogistic reasoning as follows.

1. *Interpretation.* Euler diagrams that are externally given to reasoners make explicit the *relational* information contained in categorical sentences.
2. *Inference.* Then the process of combining premise information to draw a valid conclusion can be replaced by the process of manipulating diagrammatic objects and extract the relevant relational information.

In what follows, we will concentrate on the inferential aspect in (ii). We start by explaining the representation system of Euler diagrams used in the experiment of Sato et al. [26, 27], called the EUL system. The formal properties of this system are studied in Mineshima et al. [19]. The exposition in this section is informal. More technical material as well as a detailed discussion on the motivation behind the relational approach to formalizing Euler diagrams can be found in Mineshima et al. [19].

[13] See Łukasiewicz [16], Corcoran [6], Smiley [33], and Moss [21] for such proposals.

[14] For a proof, see Mineshima et al. [20]. Conversely, all the proofs in GS that have syllogistic formulas in premises and conclusion can be simulated in categorical syllogism. This means that categorical syllogisms are *faithfully* embeddable into GS. In other words, although GS is more expressive than categorical syllogism, the syllogistic fragment of GS proves all and only the valid inferences in categorical syllogism.

The EUL system is a simple representation system; diagrams are composed only of circles and points and no syntactic device to express negation, such as "shading" in Venn diagrams, is introduced. Following traditional Euler diagrams, the EUL system represents quantificational sentences in terms of the spatial relationships between circles, in particular, inclusion and exclusion relations (see Fig. 1 in Section 1). In what follows, we refer to diagrams in the EUL system simply as Euler diagrams.

In the EUL representation system, an Euler diagram D is abstractly defined as a set of relations holding between objects in D. Based on this idea, a proof system for Euler diagrams, called GDS, is developed in Mineshima et al. [19]. An alternative, standard approach to formalization of diagrams is a "region-based" approach, where diagrams are defined as a set of regions (e.g. Shin [31]; Hammer [11]).[15] In our approach, there are three kinds of relations to be distinguished:

(i) a circle or a point X is located inside a circle A, symbolically written as $X \sqsubset A$;
(ii) a circle or a point X is located outside a circle A, written as $X \vdash\!\dashv A$;
(iii) a circle A and a circle B partially overlap each other, written as $A \bowtie B$.

In this symbolic notation, we use the same binary symbols as in GS for the relations in (i) and (ii). Indeed, the abstract representations of diagrams can be naturally translated into formulas of GS.

A deductive reasoning task generally requires us to combine the information contained in premise sentences. Given a correspondence between Euler diagrams and categorical sentences, such a process of combining the premise information can naturally trigger the process of unifying premise diagrams and extracting the relational information. We will explain, by some typical examples, how our Euler diagrams can be used in representing and reasoning about categorical sentences.

First, consider the case of the syllogism of the form: *All A are B, No C are B; therefore No C are A*.

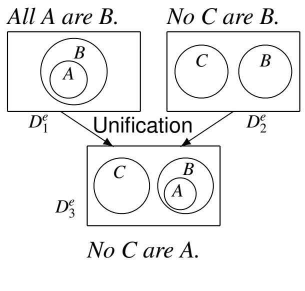


Fig.4 Solving a syllogism with Euler diagrams

Here the premise *All A are B* is associated with diagram D^e_1, where the relation $A \sqsubset B$ holds, and the premise *No C are B* is associated with diagram D^e_2, where the relation

[15] A comparison of these two approaches from a logical point of view is found in Mineshima et al. [18], where the region-based inference system is formalized as resolution calculus, in contrast to the relation-based system formalized as a natural deduction system.

$C \vdash\dashv B$ holds. These diagrams make explicit the relational information contained in the premise sentences. The operation of combining two diagrams D_1^e and D_2^e in Fig. 4 is an instance of an application of the *unification* rule.[16] In this case, the unification process consists in identifying circle B and keeping all the relations holding on the premise diagrams. The resulting diagram, D_3^e, has three relations: $A \sqsubset B$, $C \vdash\dashv B$ and $A \vdash\dashv C$. The first two are inherited from the premise diagrams D_1^e and D_2^e, and the last one, the exclusion relation $A \vdash\dashv C$, is created as a by-product of the unification process. As is seen in Fig.4, this new relation $A \vdash\dashv C$ corresponds to the sentence *No C are A*, and hence, one can arrive at the valid conclusion of this syllogism.

An important characteristic of the unification process is that by combining the two premise diagrams, one can almost automatically determine the semantic relation holding between the objects in question, without any additional operation. Such information that is automatically inferred from the result of a diagrammatic operation is what Shimojima [32] calls a "free ride".

For the process of unifying diagrams, there are two constraints that determine the spatial relationship between objects in the conclusion diagrams. Namely, for any circle or point X and for any circle Y and Z,

(C1) if X is inside Y in one diagram D_1 and Y is inside Z in another diagram D_2, then X is inside Z in the combined diagram $D_1 + D_2$;
(C2) if X is inside Y in one diagram D_1 and Y is outside Z in another diagram D_2, then X is outside Z in the combined diagram $D_1 + D_2$.

In the example in Fig. 4, the relation $A \vdash\dashv C$ is obtained using (C2). Note that these two constraints have counterparts in inference rules in GS: (C1) corresponds to the $(\sqsubset)$ rule and (C2) to the $(\vdash\dashv)$ rule.

The constraints (C1) and (C2) seem so natural and intuitive that even users who do not have explicit training on diagrammatic reasoning can exploit them to draw a correct conclusion without much effort. Theoretically, the inference rules $(\sqsubset)$ and $(\vdash\dashv)$, which are crucial for deriving valid syllogisms, are simulated in terms of the spatial constraints, (C1) and (C2). Such a simulation can happen in actual syllogistic reasoning with external diagrams. For example, a procedure using the $(\vdash\dashv)$ rule, which licenses us to derive $A \vdash\dashv C$ from $A \sqsubset B$ and $B \vdash\dashv C$, can be made manifest by perceiving the spatial relationships between diagrammatic objects as seen in Fig. 4. We can then argue that sentential (linguistic) premises themselves do not provide untrained reasoners with specific procedures of solving syllogisms in terms of $(\sqsubset)$ and $(\vdash\dashv)$, such as the ones we saw in the last section; by contrast, Euler diagrams externally given provide the reasoners with a concrete problem-solving procedure based on intuitive understanding of such constraints as (C1) and (C2).

As a second example, let us look at a syllogism having no valid conclusion, which is known to be particularly difficult for untrained reasoners (cf. Evans et al. [8]) and hence deserves special attention.

[16] The rule of unification plays a central role in the inference system for Euler diagrams developed in Mineshima et al. [20]. The system has another rule called *deletion* rule, which allows to delete an object from a given diagram. For discussion on the relevance of deletion rule to the cognitive process of information extraction, see Sato, Mineshima and, Takemura [28].

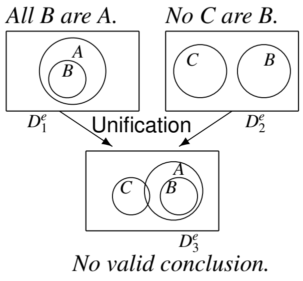


Fig.5 Solving a syllogism with no valid conclusion using Euler diagrams

In the syllogism in Fig.5, sentence *All B are A* is associated with diagram D^e_1, where the relation $B \sqsubset A$ holds, and sentence *No C are B* is associated with diagram D^e_2, where the relation $C \vdash\!\dashv B$ holds. Again, by unifying these two diagrams, one can obtain the conclusion diagram D^e_3. Note that in this case, neither constraint (C1) nor (C2) can be applied. That is, none of the the inclusion and exclusion relations between circles A and C (i.e., $A \sqsubset C$, $C \sqsubset A$, and $A \vdash\!\dashv C$) is inferable from the information conveyed by the two premises. In such a case, one needs to put circles A and C in such a way that they partially overlap each other, that is, $A \bowtie C$ holds. Note that such a convention of partially overlapping circles is common to Venn diagrams; it enables us to handle partial or indeterminate information in a relatively simple way.

To be more specific, the relevant rule is the following. For any circles X and Y,

(C3) if none of the relations $X \sqsubset Y$, $Y \sqsubset X$, or $X \vdash\!\dashv Y$ holds in the combined diagram, put X and Y in such a way that $X \bowtie Y$ holds.

Using this rule, one can see that the relations holding on the conclusion diagram D^e_3 in Fig. 5 are $B \sqsubset A$, $B \vdash\!\dashv C$, and $A \bowtie C$. The fact that $A \bowtie C$ holds in the conclusion diagram indicates that no specific semantic information about terms A and C can be drawn from the premises. This amounts to saying that there is no valid conclusion with respect to A and C (except trivial ones such as $A \sqsubset A$) in this syllogism. Here again, we can see that Euler diagrams associated with sentential premises play a dual role in the process of checking the *invalidity* of a syllogism: first, they make explicit the relational information underlying categorical sentences; second, the unification of premise diagrams using the constraints (C1) and (C2) leads us to understanding what relational information can be obtained in a given inference; when no particular inclusion or exclusion relation is newly introduced by the unification, that is, when the situation is as described in (C3), the reasoner can conclude that there is no valid conclusion of the inference.

The procedure of checking invalidity of inferences sketched here is remarkably distinguished from the standard procedure in model-theoretic semantics, according to which an inference is judged to be invalid if one can construct a counter-model in which all the premises are true but the conclusion is false. Note that some existing proposals using diagrams are also based on such an idea of counter-model constructions. Thus in Lewis Carroll's version of logic diagrams [5], an inference is invalid if it is impossible to superpose all the premises and the *negation* of the conclusion; see Lear [15] for a discusion. The diagrammatic procedure based on (C3) is distinctive in that it does not depend on any process of negating the conclusion; the information that there is no valid conclusion with respect to the two terms in question can be obtained in a direct way, via a process of unifying premise diagrams. An interesting point to note is that the process of manipulating premise diagrams, more specifically, the process of unification, is common to the tasks of checking validity and invalidity. In other words, not only *proof* but also *refutation* is realized as a syntactic process of manipulating diagrams, rather than as a process of constructing counter-models.

4 Comparisons with the mental model theory of reasoning

As mentioned in Section 1, it has been noticed in cognitive study of deductive reasoning that falsification tasks are often difficult when inference materials are presented in natural languages (cf. Evans et al. [8]). As we argued above, our logic diagrams can contribute to solving such falsification tasks by making available to users syntactic processes of unifying diagrams. An interesting feature of such unification processes is that premise diagrams themselves impose a constraint on the possible ways of unification, so that by simply trying to unify the premise diagrams, the user can observe what relations hold between the objects in the resulting diagram. In this respect, it is worth noting that there is a difference between the underlying mechanism behind unification processes of diagrams discussed in the last section and the reasoning mechanism behind the mental model theory (e.g. Johnson-Laird & Byrne [13]), which is a dominant model of linguistic (sentential) deductive reasoning in cognitive psychology.

According to the mental model theory, mental models are made up of tokens (i.e., elements of a set) and supposed to represent states of the world (cf. e.g., Bara, Bucciarelli, & Lombardo [1]). For instance, sentence *All A are B* corresponds to a model in which each token of set *A* is connected to a token of set *B*. Similarly for sentence *All B are C*, in which case each token of set *B* is connected to a token of set *C*. As a crucial step, such two premise models are integrated into a single mental model. In the present example, we can finally obtain a model in which each token of set *A* is connected to a token of set *C*. This model corresponds to the categorical sentence *All A are C*. Note that not all tokens of set *C* are necessarily connected to some token of set *A*. By this fact, we can confirm that *All C are A* cannot be a valid conclusion of this syllogism. In general, difficulties in drawing a valid conclusion are measured by the number of models that can be constructed from the integrated model.

As is suggested by this brief exposition, there is a certain similarity between processes of solving syllogisms using Euler diagrams and reasoning processes with mental models. Specifically, a unification process of Euler diagrams is very similar to an integration process of mental models, and both processes play a crucial role in deriving a valid conclusion from given premises.

However, by taking a closer look at processes of invalidity judgements, we can find that an important difference exists between the two conceptions. In the case of syllogistic reasoning by mental models, processes of integrating mental models can be performed without determining the relation between the tokens in question. That is, alternative models are to be searched for *after* performing the process of integrating the premise models. Thus, according to the theory presented by Bucciarelli and Johnson-

Laird [3], the process of constructing alternative models from an integrated model is constrained by a representational convention such as [a]. Bracketed token [a] indicates that the set containing it is represented by this individual; no new tokens can be added to the sequence with bracketed tokens. On the other hand, in the case of tokens without a square bracket, new tokens can be added so that alternative models are constructed. As an illustration, consider the case of a syllogism having no valid conclusion, as shown in Table 1 (cf. Bucciarelli and Johnson-Laird [3], p. 260). Here, there are two premise models corresponding to *All A are B* and *All C are B*. In the integrated model, then, the relationship between set A and C is indeterminate; the integration process does not require us to resolve such indeterminacy, in sharp contrast to the case of unification in diagrammatic reasoning. Processes of adding the token "b" to this integrated model are performed after the integration process. As a result, the integration process itself does not constrain the ways of constructing alternative models; see Stenning & Oberlander [35] for a related discussion.

Table 1 Representations by mental model theory for the syllogistic task from premises *All A are B* and *All C are B* to the conclusion that there is no valid conclusion.

[a] b [a] b	[c] b [c] b	[a] b [c] [a] b [c]	[a] b b [c] [a] b [c]	[a] b b [c] [a] b b [c]
All A are B	*All C are B*			
1st premise	*2nd premise*	*Integrated model*	*Alternative model 1*	*Alternative model 2*

On the other hand, the unification process of Euler (EUL) diagrams forces a user to decide what relation holds between the terms in question. That is, the configurations of diagrams constrain what relations (i.e., $\sqsubset$, $\dashv\vdash$, or $\bowtie$) are created in unifying the premise diagrams. As is well known, such a characteristic of diagrammatic representations is called *specificity* by Stenning and Oberlander [35]. In the case of syllogistic reasoning with our Euler diagrams, the relevant constraints are (C1) and (C2); these constraints are almost self-evident given intuitive understanding of inclusion and exclusion relations. In the case of falsification tasks, in particular, such a constraint can increase the chance of finding that the indeterminacy relation $\bowtie$ holds between the relevant objects, say, A and C, that is, none of the relations $A \sqsubset C$, $C \sqsubset A$, or $A \dashv\vdash C$ hold in the unified diagram.

It should be noted here that diagrams themselves are not models in the sense of model theory but certain syntactic representations that are subject to model-theoretic (set-theoretic) interpretations. In particular, if $A \bowtie C$ holds in a unified diagram, one can readily construct a counter-situation to any of the relations $A \sqsubset C$, $C \sqsubset A$, and $A \dashv\vdash C$. We may say that the *potential* to construct alternative models from diagrams are presupposed in the process of unification, in particular, in the process of entertaining a diagram containing the $\bowtie$-relation. It has been observed that the specificity of diagrammatic representations often impedes reasoning; in particular, Shimojima [32] aptly characterizes such a negative aspect of representations in terms of the notion of *over-specificity*. Interestingly, in the present case, the specificity of diagrams has a *positive* effect on the process of checking the invalidity of a syllogistic inference. That is, the failure to apply constraints such as (C1) and (C2), i.e., the failure to create a meaningful relation ($\sqsubset$-relation or $\dashv\vdash$-relation), can trigger the recognition that a given set of premises does not have a (non-trivial) valid conclusion.

We can summarize that in the case of reasoning with Euler diagrams, the process of entertaining alternative possibilities is presupposed, and implicitly triggered, in the process of unifying premise diagrams, whereas in the case of reasoning with mental models, such a process is only conducted after the process of integration, without appealing to visual constraints. In this respect, the two conceptions of combining the premise information stand in striking contrast to each other.

5 Concluding remark

In the cognitive psychology of deduction, it has long been known that solving categorical syllogisms is a difficult task for those who are untrained in logic (cf. [8]). The experimental results in Sato et al. [26] were consistent with this traditional view in that they show the performance of the Linguistic group as much lower than that of the Euler and Venn groups. The question we asked is: how can diagrams that are externally provided improve the performance of syllogism solving even for untrained people? To answer this question, in Section 1, we distinguished between interpretational and inferential efficacy of diagrams in the overall process of solving syllogisms. Now, given the relational analysis of categorical syllogisms and Euler-style diagrammatic inferences presented so far, we can elaborate and summarize the distinction in the following way.

First, concerning the interpretational side, the relational semantic information associated with quantificational sentences is often not directly accessible to reasoners. Thus, there is a tendency to interpret the sentence *All A are B* as equivalent to *All B are A*, and *Some A are not B* as implying *Some B are not A* (called *conversion error* in Sectoin 1). Euler and Venn diagrams can then help reasoners realize the underlying semantic relations implicit in categorical sentences in virtue of their spatial properties, more specifically, in virtue of inclusion and exclusion relations between objects. Hence, such external representations allow us to fix the intended relational interpretations of categorical sentences in syllogistic reasoning tasks, resulting in interpretational efficacy.

Second, concerning the inferential side, the manipulation of diagrams in the inferential process is triggered without effort, if the spatial relations holding on external diagrams are governed by *natural* constraints, i.e., constraints that depend solely upon spatial properties of diagrams and hence are accessible even to untrained users. Furthermore, given the fact that a deductive reasoning task in general requires the reasoner to assemble the information contained in the premises, the syntactic manipulations of diagrams could be spontaneously triggered when those diagrams are externally presented. The essential steps involved in the manipulations of Euler diagrams are unification processes, that is, those processes in which the inclusion and exclusion relations between objects in the unified diagrams are effectively determined using the constraints (C1), (C2), and (C3). Such a unification process is composed of steps in matching an object (a circle or a point) with another object and determining the diagrammatical relationships between the other objects. Users can exploit the natural constraints of diagrams and extract the correct procedure to apply from Euler diagrams themselves.

If these claims are correct, it would be expected that any diagram that can make explicit the relational information of a categorical sentence in a suitable way would be effective in supporting syllogistic reasoning. Sato and Mineshima [29] examines the case of a linear variant of Euler diagrams, where set-relationships are represented by one-dimensional lines, rather than by circles in a plane. The experimental results obtained there indicated that the linear diagrams for syllogistic reasoning work as effectively as Euler diagrams. This provides partial evidence that the effectiveness of external diagrams in syllogistic reasoning does not depend upon particular shapes such as circles that are specific to Euler diagrams. Rather, what is crucial is the fact that diagrams can effectively represent relational structures and aid reasoning about them.[17]

Such a comparison between reasoning with various forms of diagrams would provide further evidence to specify the semantic primitives of sentences used in reasoning tasks, and thus contribute to making progress in understanding the nature of both linguistic and diagrammatic inferential processes in human deductive reasoning. There are various ways of extending our basic fragment of syllogistic logic; e.g., relational syllogisms [24], syllogisms involving proportional quantifiers like *most* [21], and syllogisms involving conjunctive and disjunctive terms [23]. Applications of our framework to such extended syllogistic and diagrammatic inferences are left for future research.

References

1. Bara, B.G., Bucciarelli, M., & Lombardo, V. (2001). Model theory of deduction: A unified computational approach. *Cognitive Science*, 25, 839–901.
2. Barwise, J. & Cooper, R. (1981). Generalized quantifiers and natural language. *Linguistics and Philosophy*, 4, 159–219.
3. Bucciarelli, M. & Johnson-Laird, P.N. (1999). Strategies in syllogistic reasoning. *Cognitive Science*. 23(3), 247–303.
4. Calvillo, D.P., DeLeeuw, K. & Revlin, R. (2006). Deduction with Euler circles: Diagrams that hurt. In *Proceedings of Diagrams 2006*, *LNAI 4045* (pp.199–203), Heidelberg: Springer.
5. Carroll, L. (1896). *Symbolic Logic*. New York: Dover.
6. Corcoran, J. (1974). Aristotle's natural deduction system. In J. Corcoran (ed.), *Ancient Logic and its Modern Interpretations* (pp. 85–131), Dordrecht: Reidel.
7. Euler, L. (1768). *Lettres à une Princesse d'Allemagne sur Divers Sujets de Physique et de Philosophie*. Saint-Pétersbourg: De l'Académie des Sciences.
8. Evans, J.St.B.T., Newstead, S.E. & Byrne, R. (1993). *Human Reasoning: The Psychology of Deduction*. Hove: Lawrence Erlbaum.
9. Gergonne, J. D. (1817). Essai de dialectique rationelle. *Annuales de Mathematiques pures et appliqukes*, 7, 189–228.
10. Gurr, C.A. (1999). Effective diagrammatic communication: syntactic, semantic and pragmatic issues. *Journal of Visual Languages & Computing*, 10(4), 317–342.
11. Hammer, E. (1995). *Logic and Visual Information*. Stanford, CA: CSLI Publications.
12. Hammer, E. & Shin, S. (1998). Euler's visual logic. *History and Philosophy of Logic*, 19, 1–29.

[17] This claim is consistent with Gurr's [10] "well matchedness" theory.

13. Johnson-Laird, P.N., & Byrne, R. (1991). *Deduction*. Hillsdale, NJ: Erlbaum.
14. Larkin, J. & Simon, H. (1987). Why a diagram is (sometimes) worth 10,000 words. *Cognitive Science*, 11, 65–99.
15. Lear, J. (1980) *Aristotle and Logical Theory*. Cambridge, UK: Cambridge University Press.
16. Łukasiewicz, J. (1957). *Aristotle's Syllogistic: From the Standpoint of Modern Formal Logic*, Second edition, Oxford: Oxford University Press.
17. Mineshima, K., Okada, M., Sato, Y & Takemura, R. (2008). Diagrammatic reasoning system with Euler circles: theory and experiment design. In *Proceedings of Diagrams 2008*, *LNAI 5223* (pp.188–205), Berlin, Heidelberg: Springer.
18. Mineshima, K., Okada, M., & Takemura, R. (2010). Two types of diagrammatic inference systems: Natural deduction style and resolution style. In *Proceedings of Diagrams 2010*, *LNAI 6170* (pp. 99–114), Berlin, Heidelberg: Springer.
19. Mineshima, K., Okada, M., & Takemura, R. (in press-a). A diagrammatic reasoning system with Euler circles. *Journal of Logic, Language and Information*, to appear.
20. Mineshima, K., Okada, M., & Takemura, R. (in press, b). A generalized syllogistic inference system based on inclusion and exclusion relations. *Studia Logica*, to appear.
21. Moss, L.S. (2008). Completeness theorems for syllogistic fragments. In F. Hamm & S. Kepser (eds.), *Logics for Linguistic Structures* (pp.143–173), Berlin : Mouton de Gruyter.
22. Newstead, S.E. & Griggs, R. (1983). Drawing inferences from quantified statements: a study of the square of opposition. *Journal of Verbal Learning and Verbal Behavior*, 22, 535–546.
23. Nishihara, N. & Morita, K. (1988). An extended syllogistic system with conjunctive, disjunctive and complementary terms, and its completeness proof (in Japanese). *Trans. IEICE Japan*, Vol. J71-D, No.4. 693–704, 1988.
24. Pratt-Hartmann, I. & Moss, L.S. (2009). Logics for the relational syllogistic. *Review of Symbolic Logic*, 2, 647–683.
25. Rizzo, A. & Palmonari, M. (2005). The mediating role of artifacts in deductive reasoning. In *Proceedings of 27th Annual Conference of the Cognitive Science Society* (pp. 1862–1867).
26. Sato, Y., Mineshima, K., & Takemura, R. (2010a). The efficacy of Euler and Venn diagrams in deductive reasoning: Empirical findings. In *Proceedings of Diagrams 2010*, *LNAI 6170*, (pp. 6–22), Berlin, Heidelberg: Springer Verlag.
27. Sato, Y., Mineshima, K., & Takemura, R. (2010b). Constructing internal diagrammatic proofs from external logic diagrams. In *Proceedings of the 32nd Annual Conference of the Cognitive Science Society* (pp. 2668–2673). Austin, TX: Cognitive Science Society.
28. Sato, Y., Mineshima, K., & Takemura, R. (2011). Interpreting logic diagrams: a comparison of two formulations of diagrammatic representations. In *Proceedings of the 33rd Annual Conference of the Cognitive Science Society* (pp. 2182–2187). Austin, TX: Cognitive Science Society.
29. Sato, Y., & Mineshima, K. (2012). The efficacy of diagrams in syllogistic reasoning: A case of linear diagrams. *Proceedings of Diagrams 2012*, to appear.
30. Scaife, M. & Rogers, Y. (1996). External cognition: how do graphical representations work? *International Journal of Human-Computer Studies*, 45, 185–213.
31. Shin, S.-J.(1994). *The Logical Status of Diagrams*. New York: Cambridge University Press.
32. Shimojima, A. (1996). *On the Efficacy of Representation*. PhD thesis, Indiana University.
33. Smiley, T. (1974). What is a syllogism? *Journal of Philosophical Logic*, 1, 136–154.
34. Stapleton, G. (2005). A survey of reasoning systems based on Euler diagrams. *Proceedings of Euler Diagrams 2004*, *ENTCS 134* (pp. 127–151). Amsterdam: Elsevier.
35. Stenning, K., & Oberlander, J. (1995). A cognitive theory of graphical and linguistic reasoning. *Cognitive Science*, 19, 97–140.
36. Stenning, K., Cox, R. & Oberlander, J. (1995) Contrasting the cognitive effects of graphical and sentential logic teaching: reasoning, representation and individual differences. *Language and Cognitive Processes*, 10, 333–354.

Visualizing Syllogisms: Category Pattern Diagrams versus Venn Diagrams

Peter C-H. Cheng

Department of Informatics, University of Sussex, Brighton, UK
p.c.h.cheng@sussex.ac.uk

Abstract. A new diagrammatic notation for Syllogisms is presented: *Category Pattern Diagrams, CPDs*. A CPD configures different styles of line segments to simultaneously assign quantification values to categorical variables and relations among them. The design of CPDs attempts to coherently visualize the structure of syllogisms at various conceptual levels. In comparison to Venn Diagrams and conventional verbal expressions of syllogisms, the potential benefits of CPD may include: a relatively straightforward inference method; simple rules for evaluating validity; applicability to multiple (>2) premise syllogisms.

1 Introduction: the project and programme

This paper is part of a project that is attempting to develop a novel set of related diagrammatic notations for various systems logic. The project's objective is to design a family of diagrammatic notations that share a common representational scheme for encoding logical states of affairs and a common inference method. The focus here is syllogisms, with the introduction of *Category Pattern Diagrams*, CPDs. The design of CPDs builds directly upon our previous work on *Truth Diagrams* for propositional logic [1,2]. In turn, CPDs are being used as intermediate stage to develop a related notational system for full predicate calculus. The overarching aim of the project is to show how re-codifying systems of logic in closely related notational systems may reveal the similarities and differences in the conceptual structures of those logics.

The project is, in turn, part of a larger *Representational Epistemic* research programme that is studying how notational systems encode knowledge and the potential cognitive benefits that novel codifications of knowledge may confer on higher forms of thinking [3-6]. The core principles of the Representational Epistemic approach address how to design representational systems for knowledge rich topics. They claim that directly encode the fundamental conceptual structure of a topic in coherent notational schemes will provide semantically transparency and thus enhance problem solving and conceptual learning in multiple ways [3-5]. Previous knowledge domains that have been re-codified as part of the programme include electricity, probability

3rd International Workshop on Euler Diagrams, July 2, 2012, Canterbury, UK.

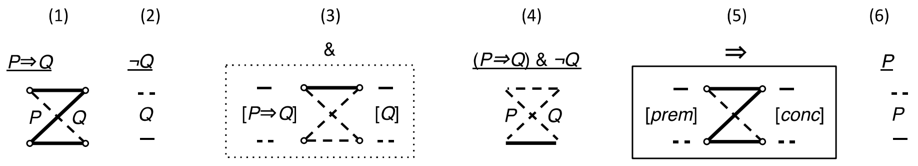


Fig. 1. Truth Diagram demonstration of the validity of Modus Tollens

theory and (school) algebra [3-5]. The project on logic is extending the scope of the programme by providing further stringent test cases for the Representational Epistemic claims.

The previous work in the logic notation design project developed *Truth Diagrams*, TDs, to re-codify propositional logic (and Boolean Algebra) [1,2]. Fig. 1 shows an example in which the validity of Modus Tollens is demonstrated. The details of TDs are not essential to consider here, rather it is the overall form of the notation that is of concern, because the design of CPDs aims to adopt a similar representational scheme and inference method. In TDs letters are labels for variables and configurations of line segments assign truth-values to propositional variables and relations among those variables. Solid lines stand for True and dashed for False. Fig. 1.1, 1.2, 1.4 and 1.6 are unary or binary relations of variables involving *P* and *Q*. The inference method creates a composite diagram, Fig. 1.4, by combining the premise diagrams, Fig. 1.1 and 1.2, using a diagrammatic operator, Fig. 1.3, which specifies the types of the lines to draw in the composite diagram given the permutations of line types in the premise diagrams. The validity of the inference is determined by comparing the structure of the composite diagram with the diagram for the given conclusion, Fig. 1.6, using a simple set of diagrammatic validity rules, Fig. 1.5, which specifies correct correspondences between the types of lines in the two diagrams. TDs constitute an efficient method to reveal how the propagation of patterns of truth-values determines the structure and validity of interferences. Taken together, the structure of the diagrams, the composition operators and the validity rules provide a novel, complete and sound, system that reveals conceptual structures (symmetries and regularities) on multiple levels that are typically hidden by standard formula notation [1,2].

The specific aims of this paper are: (a) to introduce *Category Pattern Diagrams*, *CPDs*, as a notation for syllogistic reasoning, that adopts a similar representational scheme and inference method to TDs; (b) to compare CPDs with syllogistic inferences using Venn diagrams and the traditional verbal approach; (c) to examine how codifications of syllogisms in these alternative notational systems provides quite different perspectives on the underlying conceptual structure of syllogisms, with varying degrees of coherence, and the impact this has on the ease of making inferences. Thus, the paper has the following sections: 2 is a brief reminder about syllogisms; 3 describes the graphical structure of CPDs; 4 gives the procedures of composing premise diagrams in to a result diagram; 5 provides the method to determine whether the result diagram correctly implies the given conclusion diagram; 6 extends the approach to multi-premise syllogisms, sorites; and, 7 discusses the overall efficacy of the CPD encoding of syllogisms and considers implications for the design of notations to encode logic.

2 Syllogisms: a brief reminder

See [7] and [9], for example, for full introductions to syllogisms; but as a reminder, consider syllogisms S1 and S2.

S1. No diagrams are sentential notations
<u>All Venn Diagrams are diagrams</u>
No Venn Diagrams are sentential notations

S2. All Category Pattern Diagrams (CPDs) are diagrams
All diagrams are effective representations
No effective representations are poor systems for learning
<u>Some poor systems for learning are sentential notations</u>
No CPDs are sentential notations

S1 is a classical two-premise syllogism, consisting of a *major premise*, a *minor premise* and a *conclusion*. The *middle* term, *M*, occurs in both premises and the *predicate* and *subject*, *P* and *S*, are the *major* and *minor* terms of the major or the minor premises, respectively. (M is a subject in the major premise and a predicate in the minor premise.) The *quantity* and *quality* of S1's *major* premise happens to be *universal* and *negative* (*No M are P*); such propositions are labelled 'E'. S1's *minor* premise is universal and affirmative (*All S* are *M*); labelled 'A'. The conclusion is also an E proposition (*No S are P*). *Particular affirmative* propositions are labelled 'I' and *particular negative* propositions labelled 'O'. The *mood* of a syllogism is its particular permutation of proposition types for the two premises and the conclusion: S1's mood is EAE. *S* always precedes *P* in syllogism conclusions. The four possible permutations of the order of the premise variables are called *Figures*; S1 is of Figure type 1: *M-P*, *S-M*. The Mood and Figure type of S1 may be summarised as 'EAE-1' and like all valid syllogisms has been given a name, "Celarent" [1,2].

To determine the validity of syllogisms in verbal form, one may apply five rules concerning the quality and quantity of the propositions. The quality rules state: (QL1) no conclusion may follow when both premises are negative; (QL2) a conclusion is negative when either premise is negative; (QL3) a negative conclusion cannot follow from two affirmative premises. The quantity rules rely upon the notation of *distribution*, which is the extent to which all the members of a category are affected in a proposition [7]; e.g., *S* is distributed in *All S are P*, but variable *P* is not. The quantity rules state: (QN1) the middle term must be distributed in one or both premises; (QN2) if a term in the premises is not distributed, then it must not be distributed in the conclusion. These rules are challenging to understand and apply, and explanations of why they govern the validity of inferences are not straightforward to give.

Venn Diagrams, e.g. Fig. 2, provide a more comprehensible means to assess the validity of syllogisms. First, a diagram is drawn with three fully intersecting circles to represent all the possible combinations of sub-sets, Fig. 2.3. Then, beginning with any universal premises, Fig. 2.1, corresponding regions in Fig. 2.3 are shaded for empty sets. Subsequently, for any particular premises, Fig. 2.2, a cross is drawn in any corresponding non-shaded region of Fig. 2.3. Care is needed to correctly locate

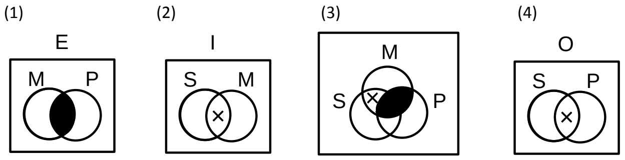


Fig. 2. Venn Diagram for the Ferio syllogism

the shading and crosses to take into account the term not mentioned in each premise. The inference is valid if the conclusion, Fig. 2.4, can be read directly from the pattern of shading and crosses in the three-circle diagram.

S2 is a sorites, a multiple premise syllogism. Their general form is P_1–P_2, P_2–P_3, …, P_{n-1}–$P_n \Rightarrow P_1$–P_n. The particular form of S2 is: *All C are D, All D are E, No E are P, Some P are S ⇒ No C are S*. Although Venn Diagrams can be systematically drawn for four and more sets [8], the diagrammatic benefits of that approach appear to be reduced for larger numbers of premises.

From this brief overview, it is clear that to re-codify syllogisms CPDs must do many things: identify the categorical variables; denote whether things belong to each category or not; specify relations among the variables, i.e., the mode and Figure; signify the quantification of the multiple subsets defined by those relations; have the potential to represent multiple premises (>2); provide a method to infer the quantity values of variables in the combined relations; establish a procedure to determine whether inferences correctly imply the given conclusion.

3 Graphical structure of CPDs

Fig. 3 shows examples of CPDs for unary, binary and ternary relations of categorical variables. Each diagram represents a state of affairs relating the categories identified by the letters. The letter is a label for a category and the lines run between the specific positions relative to the letters. In a unary variable diagram, Fig. 3.1, horizontal line segments are positioned above and below the letter. For the binary relation, Fig. 3.2, four lines run between the letters with their ends located at the four possible combinations of positions above or below each letter. We will call such line segments *connectors*. In the ternary relation diagram, Fig. 3.3, the eight connectors are composed of two binary connectors joined near the middle letter and with free ends associated with the other two letters. The eight connectors are arranged as four pairs: (1) an inverted triangular pattern; (2) an upright triangular pattern; (3) a descending parallelogram pattern, which slopes downwards from left to right; (4) an ascending parallelogram pattern. As will be seen below, this design of the ternary CPDs is intended

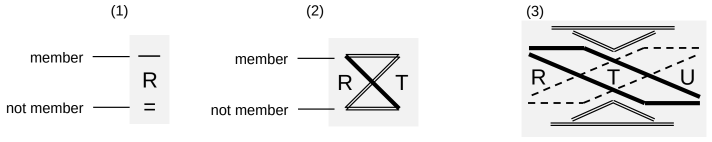


Fig. 3. Unary, binary and ternary Category Pattern Diagrams

to support judgments about the validity syllogisms.

The position of the ends of a connector, top or bottom, refers to possible membership or possible non-membership of the category, respectively. Fig. 3.1 and 3.2 include labels making this explicit. Each connector is a particular *case*, a combination of inclusion or exclusion of things in the subsets of the variables of the relation. The unary relation has two cases and the binary relation has four cases.

Consider examples of some cases in the ternary CPD of Fig. 3.3. In the top inverted triangle the upper straight connector refers to the case of the membership of all variables, where as the \/ shape connector is the case of subsets *R* and *U* membership but *T* not. In general, the local altitude of the middle point or free ends of a connector within a triangle or parallelogram indicates the membership status of a subset, with a high position in the shape for membership and low position for absence. In the descending parallelogram the ¯\ connector refers to the membership of *R* and *T* and the absence of *U*, whereas the _ connector refers the membership just of *R*.

The line style of the connector assigns a quantity to a case. There are three styles for three quantifiers: (1) a single solid connector is *some* – at least one instance of the case; (2) a dashed connector is *none* – no instance of the case; (3) a solid double-line connector means *no information* (*no-info*) – the quantification of the case is not known; it may either be *some* or *none*. In Fig. 3.1 the top some *connector* specifies that something is a member of *R* and the bottom *no-info* connector means it is not known if things are excluded from *R* or not. In Fig. 3.2 the three double-line connectors means the only specific information provided relates *R* and *not T*, and its solid single-line connector says that at least one thing is a member of R is not a member of T: in other words, the diagram reads *Some R are not T*. Consider three cases in Fig. 3.3. The double-line top connector of the upper triangle pair says that there is nothing known about the assignment of members to the intersection of *R*, *T* and *U*. In the descending parallelogram the solid line of the lower _ shape connector indicates that at least one thing is a member of *R* but it is absent from *T* and *U*. The dashed line of the /¯ connector says there is nothing that is *T* and *U* and not *R*.

Each connector in a CPD is equivalent to a region in a Venn diagram.

Fig. 4 shows CPDs diagrams for the four syllogistic propositions, A, E, I and O (and gives their verbal expressions). Notice that all the CPDs have three *unknown* connectors (double-lines) and either a single *some* or a *none* connector to constitute the particular and universal propositions. The intersection of the two sets is the top connector; the two exclusive subsets of the variables are the ascending and descending diagonals; the exclusion of both sets is the bottom connector. When the order of the terms in a proposition is swapped, the order of the letters in the CPD is simply reversed. Equivalently, the pattern of the lines may be reflected with the letter positions fixed. (If both the letters and lines are reversed, the proposition is unchanged.)
Notice that the patterns of lines in E and I are symmetrical, which has interesting implications for the validity of certain syllogisms; as will be seen below.

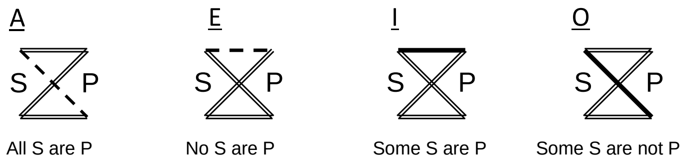


Fig. 4. The four types of syllogistic propositions as binary CPDs

Turning to ternary CPDs, Fig. 5.2 shows a generic ternary relation CPD with numbered connectors that show the corresponding regions of the Venn diagram in Fig. 5.1. Fig. 5.3 and 4 are two examples of specific ternary relations. In Fig. 5.3 the descending parallelogram says there is *nothing* that is *S and not P*, whatever the case with *M*. The ascending parallelogram says that *no-info* holds for *not S and P*, for both values of *M*. The top and bottom triangles both possess cases in which either there are no instances present or that *no-info* occurs, for different values of *M*. Fig. 5.4 shows other patterns of connectors including the assignment of *some* to one case. With a little experience, identifying individual cases in CPDs appears to be as easy as finding sub-sets in a Venn diagram. Similarly, selecting pairs of cases for the same values of *S* and *P*, as is required to judge the validity of a syllogism, also appears to be comparable in both notations. However, we will see below that the CPDs and Venn diagrams diverge when more than three propositions are considered.

That completes the overview of the syntax and semantics of relational CPDs. The next section considers how to make inferences with CPDs.

4 Composition of Binary CPDs

Fig. 6 shows the CPD for the *Celarent* syllogism (EAE-1: *No M are P*, *All S are M*, therefore *No S are P*; Figure type 1). In outline, the overall procedure for syllogistic inferences with CPDs has two stages. First, given the two premises (Fig. 6.1 & 6.3), the conjunction operator (6.2, see below for a full explanation) is applied to generate the ternary *result* diagram (6.4). (The term *result* refers to the set of implications derived from the premises as distinct from the given conclusion.) In the second stage, the pairs of connectors of the result diagram are compared to the conclusion diagram (6.5.E) to check that the result diagram fully and correctly implies the conclusion diagram. (In Fig. 6 the desired conclusion (6.5.E) is highlighted but three others are included for the discussion of invalid inferences below.) This stage compares the types of connectors in the result diagram with the corresponding conclusion connectors using a table of validity rules (6.6). This section describes the construction of the ternary result diagram and the next section gives the procedure for testing validity.

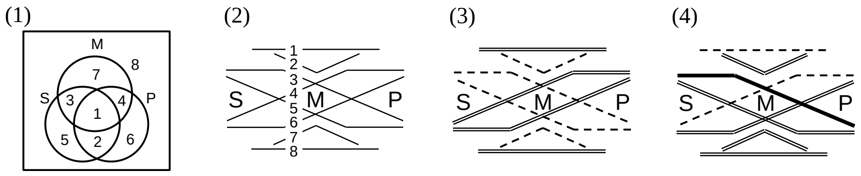


Fig. 5. Ternary CPDs

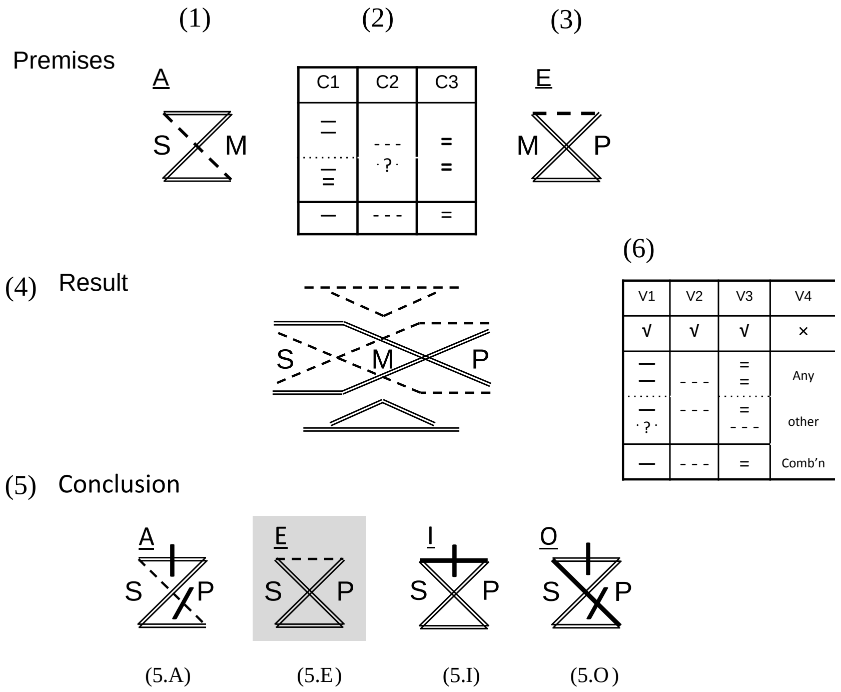


Fig. 6. EAE-1 Syllogism (and some alternative conclusions)

To construct a ternary result diagram we simply consider each of the eight connectors in the CPD in turn. Fig. 7 shows two examples of the construction of two connectors in the result CPD of Fig. 6.4. Three steps are required for each connector. Step 1 – Fig. 7, arrows 1: determine the shape and position of the new connector from the relevant connectors in the two premise CPDs. Step 2 – arrows 2: from the styles of the pair of premise connectors find the relevant composition rule. Step 3 – arrows 3: the style of the new result connector is given by the output of the selected rule.

In preparation for step 1, the two premise diagrams are drawn so that the *middle* term (*M*) will be in the centre of the new diagram and the subject term (*S*) on the left and predicate term (*P*) on the right, see Fig. 6 and 8. *M* is in the middle because it is common to both binary premise diagrams. The *S* and *P* arrangement will facilitate the comparison of the result diagram with the conclusion diagram later (see below). Fig 8.1 illustrates this process for a ternary CPD of no particular mood (faint lines for arbitrary connector types). If *S* is to the right and *P* to the left of *M* in the premise diagrams, as in Figure type 1 syllogisms such as Fig. 6, they can simply be put together without further ado. If the premises are different syllogism Figures, then one or both of the premise diagrams is reflected before they are combined; for example, in a type 2 *Figure* syllogism, the *M* term occurs on the right of both the binary premise diagrams, so just the *P*-*M* diagram needs to be reversed. Thus, all the possible moods and Figures of syllogism handled.

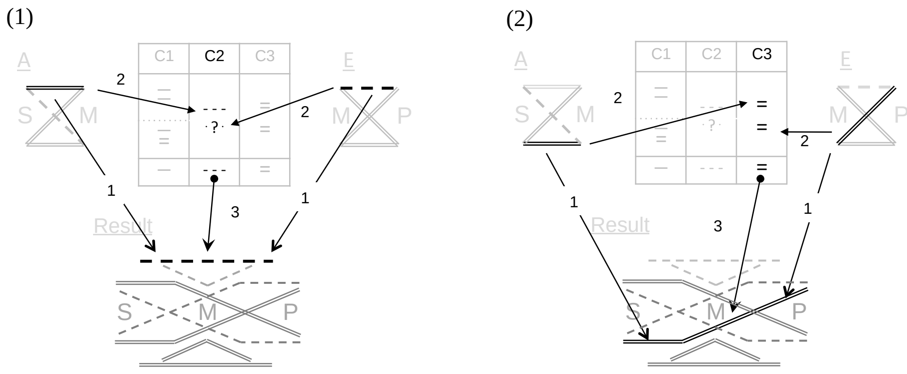


Fig. 7. Composition of ternary CPDs – connector shape and style

Now, step 1 builds each ternary result connector from possible pairs of the premise connectors. The top connector of result CPD, Fig. 7.1 (arrows 1), combines the top connectors of the premises. The _/ shape result connector, Fig. 7.2 (arrows 1), is assembled from the bottom and ascending diagonal connectors of the premises. In general, for each connector in one binary diagram there are two possible associated connectors in the other diagram. Fig. 8.2 shows how the four pairs of connectors in Fig. 3.3 and 6.4 are obtained from the binary diagrams in Fig. 8.1. Each of the four patterns in Fig. 8.2 corresponds to a particular case of *S* and *P* values, but the values of *M* differ.

In Step 2, we find the quantification value for the new connector by looking up the values of the premise binary connectors in the composition operator look up table in Fig. 9. A copy of this table is reproduced between the two binary diagrams in Fig. 6 and 7 for convenience. Given the three possible types of each of the two premise connectors, 3^2=9 permutations are possible. The table determines mappings from pairs of premise connector types at the top of each column to the result connector type at the bottom. The ‘▯?▯’ symbol in Fig. 9 means any type of connector. (C1) The result of the operator will obviously be a *some* connector when both premises are *some* connectors. Whenever one premise connector is a *some* connector and the other a *no-info* connector, the result is also a *some* connector, because just one premise possessing a member will ensure that the new case contains a member. (C2) When both of the connectors are *no-info* types, combining them provides no new information; therefore, the result is also a *no-info* connector. (C3) Given a single *none* connector, or pair of them, the result must be a *none* connector, because the presence of any members of the new combined category is forbidden. For example, in Fig. 7.1 rule C2 applies to the left *no-info* and the right *none* connectors, so the new connector will be have *none* style.

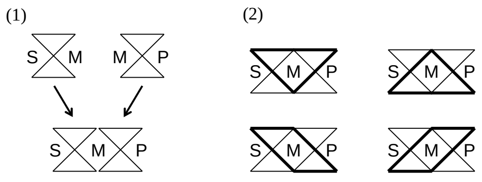


Fig. 8. Composing ternary CPDs

In step 3, we simply draw the result connector in the style given by the output of the rule selected in step 2, in the position determined in step 1; in Fig. 7.1 (arrow 3) this is a dashed top connector. In Fig. 7.2 the bottom and the ascending diagonal premise connectors will give a _/ shaped result connector (step 1), rule C3 applies because both premise connectors are both *no-info* (step 2), so the result connector is drawn in position as a *no-info* connector (step 3). Repeating the steps for the other six connectors completes the result CPD. As the two premise CPDs in Fig. 6 both possess just *no-info* or *none* connectors the resulting ternary CPD contains only connectors of these types.

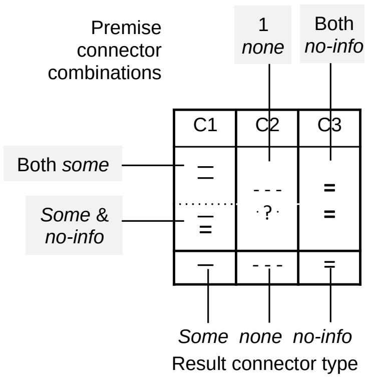


Fig. 9. Composition operator rules

5 Determining Validity

The second stage of the CPD approach compares the result diagram with the given conclusion diagram to establish whether, or not, each case of possible assignments of values of *S* and *P* in the conclusion is validly implied by the two possible cases for the same assignment of *S* and *P* in the result. As noted, the pairwise design of the connectors in the ternary CPDs supports these comparisons. Fig. 10 shows the correspondence between the pairs of result connectors and the conclusion connectors: the upper result triangle maps to top conclusion connector; the descending parallelogram to the descending connector; the ascending parallelogram to the ascending connector; the bottom triangle to the bottom connector. The subsets of *S* and *P* are the same in the result and conclusion for each matching case.

For each of these matches, we now determined whether the types of the two result connectors correctly imply the type of the conclusion connector. Fig. 11 provides a look up table for valid matches, where each column is a validity rule. (V1) If either or both of the result connector types are *some*, then the conclusion connector is *some*, because the presence of any member in the result implies the conclusion will have a member. (V2) Two *none* result connectors imply a *none* conclusion, because the total absence of any category members in the result implies an absence of members in the conclusion. (V3) Two *no-info* connecters, or one with a *none* connector, implies a *no-info* conclusion connector, because these combinations provide no information about whether there are category members or not. (V4) No other permutations of result and conclusion connectors are valid. Given that each of the three connectors may be one of three types, a total 3^3 different permutations exist, so the four rules of Fig. 11 constitute

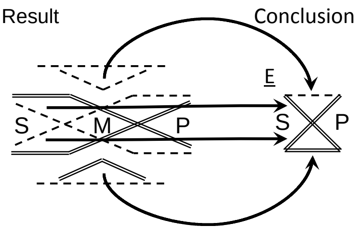


Fig. 10. Matching result connectors to the conclusion connectors

a concise encoding of the 27 possible ways result connectors may, or may not, validly imply the conclusion. Again, this conciseness may be attributed to the representation of the possible quantification values as three styles of lines.

Now applying the validity rules to Fig. 6, the *none* top connector of the target conclusion (Fig. 6.5.E) is correctly implied by the result, because the upper triangle has two *none* connectors (Fig. 6.4) – Rule V2. The descending parallelogram correctly implies the respective *no-info* descending conclusion connector, because the result parallelogram has one *no-info* connector and one *none* connector – V3. This rule applies to the ascending parallelogram in the same fashion. It also applies to the bottom triangle but in respect to the two *no-info* connectors. Therefore, as all four of the result connector pairs correctly imply their conclusion connectors, the overall inference is valid. Had just any one of these matches been invalid, the overall implication would have been invalid.

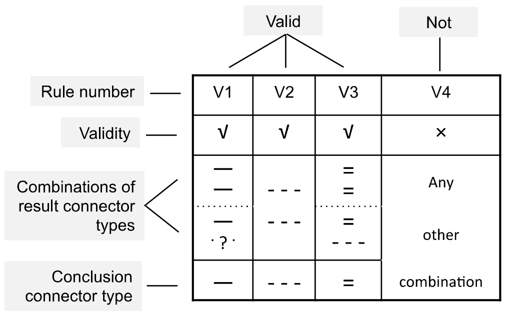


Fig. 11. Validity rules

The other types of proposition, A, I and O, are shown in Fig. 6 as alternative conclusions, which we now demonstrate are not implied by the conjunction of premises E and A; i.e., EAA-1, EAI-1 and EAO-1 are not valid syllogisms. The bars on the conclusion connectors in Fig. 6.5.A/I/O identify those that are not satisfied in the result diagram. In the case of the A conclusion, the top *no-info* connector is not implied by the pair of *none* connectors in the upper triangle (V4 true, V3 violated), and the descending *none* diagonal is not implied by a single *none* connector in the parallelogram (V4 true, V2 violated). For the I proposition the top *some* connector is not implied, because there is no *some* connector among the pair of in the upper triangle of the result (V1 violated), and similarly for the *some* descending connector in the O proposition (Fig. 6.5.O).

Fig. 12 derives the valid *Ferio*, *Festino* and *Ferison* and *Fresison* syllogisms (EIO-1, 2, 3, 4), and has three points of interest. (1) The presence of the *some* connector yields a *some* connector or a *none* connector in the result CPD when it is combined with a *no-info* or a *none* connector, respectively, from the other premise. (2) The match of the *some* connector in the result diagram and the O conclusion satisfies V3, but the A, E and I conclusions neatly show how different forms of mismatch are easily spotted. (3) Both premise diagrams are symmetric, because their only non *no-info* connectors are the top lines, which means that the overall configuration of the result ternary CPD is invariant: the orders *S*, *P* and *M* does not matter, which is why the EIO mood is the only one that is valid for all four *Figures*. By the same reasoning, this explains why valid syllogisms often occur in pairs; they have an E or I as a premise.

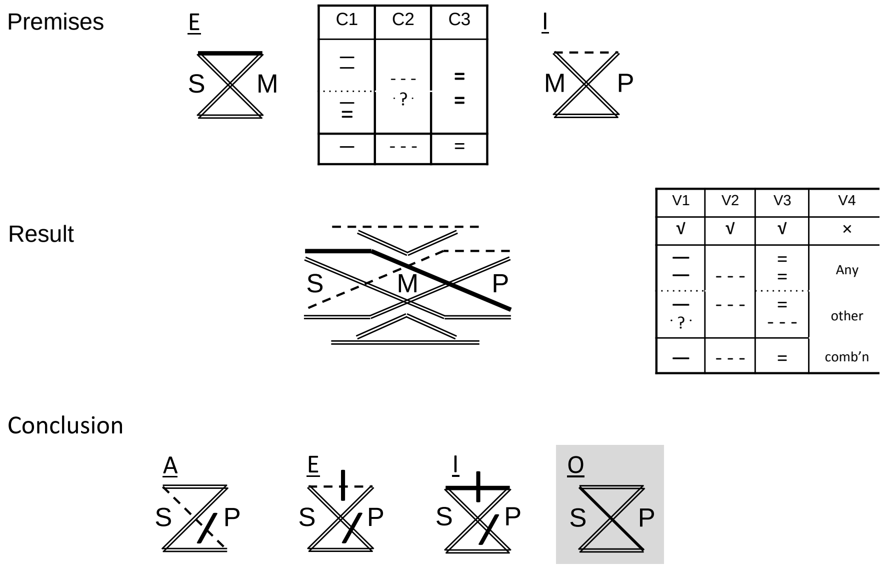


Fig. 12. EIO-1/2/3/4 syllogism (and some alternative conclusions)

Is the CPD approach for classical syllogisms complete and sound? All the 256 possible combinations of mood and figures have been examined. CPDs are complete because all 15 valid syllogisms [7] are found to be valid in the approach. It is sound because none of the 241 invalid syllogisms [7] are found to be valid. (As the composition and validity rules are few in number and simple, a spreadsheet was setup to test all 256 syllogisms en masse.)

6 Sorites

The CPD approach extends beyond classical syllogisms. Fig. 13 shows two examples of sorites, or polysyllogims. In each, the sequence of premises is on the left and the conclusion on the right. A ternary CPD has eight connectors, and as each additional proposition doubles the number of cases, quaternary and quinary CPDs will have sixteen and thirty-two connectors, respectively, so would consequently be cumbersome to draw. However, given the relative simplicity of the composition rules and validation rules it is not essential to expand the row of premise CPDs, but rather we may consider possible paths along connectors from the first variable through to the last. The composition rules in Fig. 9 may be applied iteratively to a sequence of connectors. The top row of Fig. 13.1 are all *no-info* connectors, so Rule C3 (Fig. 9) yields an overall *no-info* path. The \ _ _ shaped path has one *none* connector and two *no-info* connectors, so its overall path is *none*. Is this sorite valid? In an equivalent fashion to Fig. 10, all the paths through the premises from a specific start point to a specific end point are compared to the conclusion connector that has corresponding points; for example, paths from the top-left to bottom-right through of the sequence of

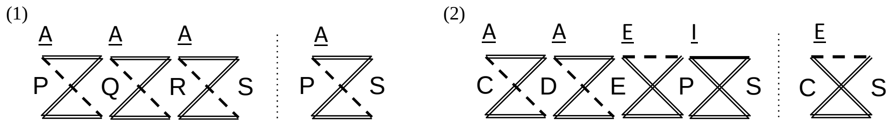


Fig. 13. Two sorites as CPDs

premises (top *P* to bottom *S*) corresponds to the descending diagonal connector of the conclusion (top *P* to bottom *S*). The rules in Fig. 11 are used to judge whether all the types of these paths correctly imply the conclusion connector type. The top connector of the conclusion of Fig. 13.1 is a *no-info* connector and by inspect we can see that all paths (‾‾‾‾, \/‾‾, _/, ‾‾\/) from the top left to the top-right of the premises are either *no-info* or *none* paths by rules C2 and C3. Thus, all four paths satisfy V3. Similarly, the ascending connector in the conclusion is correctly implied by the four paths from the bottom-left to top-right, because there is at least one *no-info* path and the rest are *none* paths, applying C3, C2 and V3. The same is true of bottom conclusion connector and the bottom-left to bottom-right premise paths. The descending line in the conclusion is a *none* connect. Again by inspection, we see that all four paths from the top-left to the bottom-right contain one or two *none* connectors, so by rule C2 all the paths are *none* paths, which means that the conclusion is correctly implied (V2 satisfied). As all the conclusion connectors are correctly implied the overall inference is valid.

Our second syllogism above, S2, is a four-premise inference with a mixture proposition types (A, E and I). The letters of the variables in Fig. 13.2 have been chosen to match the terms in the S2. Although this example is more complex than Fig. 13.1, testing its validity is relatively straightforward. Consider the top *none* connector of the conclusion. Rule V2 say that all premise paths must be the *none* type for this to be correct, however we immediately see that there is a path consisting only of *no-info* connectors, ‾‾\/, so this case is not valid, and in turn the overall inference is invalid: QED. (Testing the other cases is not arduous. All the other premises paths correspond to three *no-info* connectors in the conclusion. By inspection all the cases include at least one *no-info* path (C3) and *none* paths as the only other type (C2), so all have a mixture of *no-info* and *none* paths, therefore all three conclusion connectors are correctly implied, because the conditions for V3 are met. Nevertheless, the sorite is invalid, because the validation of top conclusion connector failed.)

This inspection method may, of course, be applied to two-premise syllogisms, and is simpler than constructing of the ternary result diagram (Fig. 6 and 12). However, the ternary result diagrams are nevertheless worth considering, because they provide an explicit introduction to the analysis of the structure for binary CPDs sequences that is needed to familiarize learners on the composition of connectors and about the matching of multiple connectors to test validity. Quaternary CPDs can be drawn with four groups of distinct patterns of four connectors that serve the same role as the four pairs of connectors in ternary CPDs. However, they are cumbersome, because they include 16 distinct lines. Clearly, higher order CPDs will be impractical to draw. Fortunately, this limitation of CPDs is mitigated by the potential to use the inspection method on linear sequences of binary CPDs for many purposes when dealing with multi-premise syllogisms.

7 Discussion

Two of the aims of developing Category Pattern Diagrams were (1) to investigate whether a new notation for syllogisms could be designed using a similar representational scheme and inference method to that devised for propositional logic Truth Diagrams (c.f., Fig. 1), and (2) to examine whether the possible benefits of the CPD notation were similar to those of the TD notation.

CPDs have been successfully developed using a scheme in which assignments of values to variables, and values to relations among variables, is based on the position, shape and style of line segments running among letters for categories. CPDs used three styles of lines for *some*, *none* and *no-info* connectors, whereas TDs have two styles for truth-values. Unlike the many diagrammatic composition operators of TDs, there is just a single composition operator for CPDs, as syllogisms merely concern conjunctions of propositions. Although there are many possible permutations of values for a pair of connectors, the CPD composition operator includes just three simple rules. Similarly, the method for testing the validity of an inference consists of just four simple rules to compare connectors in the conclusion and combined diagram of the premises. When one is new to CPDs, an explicit result diagram may be drawn in order to work methodically through all the permutations of connectors (e.g., Fig. 6 and 12). However, when one is familiar with the system, the validity of an inference may be determined by inspecting paths running through the sequence of premises (e.g., Fig. 13). This approach is feasible because (i) the small number of simple composition rules enables one to mentally compute the overall type of a path traced along successive connectors and (ii) the small number of simple validity rules means that the implications of a group paths can be readily judged in relation to a conclusion connector.

The simple rules of the CPD approach stands in marked contrast to the conventional verbal approach to the evaluation of syllogisms that relies on the three quantitative and two qualitative rules given in section 2. Because the quality rules, QL1-3, are stated in terms of negatives or even double negatives, this inevitably makes them somewhat tricky to apply. (They may be restated in positive terms, but at the cost of introducing awkward disjunctions to work through.) The same comment holds for quantity rule QN2. Further, both quantity rules are also challenging to apply, because they not only concern the distribution of terms among the premises and conclusions, but very notion of distribution is conceptually demanding to apply to all the terms in all four types of syllogistic proposition. Inferences with CPDs works at a more elemental level, with judgments about the overall validity of an inference depending upon simple comparisons of whether the assignment of values to conjunctions of variables are compatible, which is done by visually matching the styles of simple patterns of line segments.

A similar claim holds for Venn Diagrams, as the assessment of the validity of an inference revolves around whether the presence of a cross or the shading of particular region in the three circle diagram are consistent with the conclusion. Whether CPDs or Venn diagrams, in themselves, are better visualization for classical two premise syllogisms will depend on particular representation design issues. One such is the explicit representation of the absence of information in CPDs (i.e., the *no-info* connector) versus the implicit encoding in Venn Diagrams (i.e., no × and no shading). Another issue is the efficacy of representing sub-sets using spatially contained regions versus distinct line segments. Such design issues will require empirical tests with users. However, an advantage of CPDs over Venn diagrams is in relation to multi-premise syllogisms. Venn himself show how to draw his diagrams for four and more sets, but even with more simpler modern designs (e.g., [8]), the difficulty of dealing with large numbers of premises increases more rapidly for Venn Diagrams than with CPDs. The complexity of constructing the diagram and interpreting its relations appears to grow with the power of the number of sets. In CPDs the difficult arises with the growing number of paths, but this is mitigated by the multiple constraints that the construction rules and validity rules usefully offer. For example, composition rule C2 means all combinations of paths up or down stream of a *none* connector in a sequence of binary premise diagrams will be *none* paths. Finding just a single *no-info* or *some* path corresponding to a *none* connector in the conclusion invalidates the whole inference.

The comparisons of CPDs to the verbal and Venn diagrams approaches allow some observations to be made about the general nature of how notations systems might effectively codify logic. First, although both CPDs and Venn Diagrams are graphical representations they use quite different schemes to encode the same concepts, which again supports the theoretical claim that it is the nature of the relation between the conceptual structure of the ideas being encoded and the characteristics of a notational that largely determines the efficacy of a representational system [3-6]. It is not merely that a graphical representation is spatial or geometric in nature that provides potential benefits to reasoning, but how particular diagrammatic properties encode and interrelate the concepts. Although the spatial containment on the plane provides an initially compelling device to encode a small number of set memberships, the scheme becomes rather less efficacious with larger numbers of sets.

The second observation is that the composition and validity rules of CPDs operate at the "elemental" level of the assignment of fundamental quantity values (*some*, *no-info* and *none*) to the "atomic states" of member and non-membership of the subsets of variables and relations. As a consequence the basic rules of the system are simple and relatively few in number. It is therefore possible to hypothesize that the conceptual difficulties we face in order to understand syllogisms does not arise from the intrinsic nature of the topic, but rather is due to the complexity generated by combinatorics of these fundamental elements in situations with multiple terms. The design of CPDs appears to demonstrate that directly encoding the fundamental concepts of the syllogism domain in the primary representational schemes of a notational system creates an effective codification of the topic (potentially). As such, this would be a further example of the core Representational Epistemic principle, which was previously demonstrated in a range of other knowledge rich topics [3-6].

The third observation concerns how the direct encoding of the fundamental concepts supports reasoning with the new notation. It has previously been theorized such codifications of knowledge produce a semantically transparent system, in which many of the concepts at different levels of granularity, levels of abstraction and in alternative perspectives are readily accessible in the same of expressions of the notation [3-6]. It appears that this claim is also true for CPDs, as they provide explicit access to multiple types of information that are variously used to make inferences with and to explain syllogisms. These include: the identification of categories (labels); distinguishing the subsets of variables (high and low position); specification of relations among variables (connector shapes); assignment of values to the variables (positioning of connectors relative to letters); the type and order of propositions, or moods (arrangement of binary CPDs); the ordering of the variables within a proposition, or Figures (arrangement of unary CPDs in each binary CPD). As the composition operator and validity rules apply directly to patterns of connectors their effect on the categorical state of affairs tends to be plain. Further, by examining overall patterns of connectors for different combinations of mood and Figures one can gain a sense of regularities that follow from the underlying categorical constraints (e.g., the impact of the symmetry of the E and I) and also the implications of concepts such as distribution (e.g., by adding symbols to explicitly show the distributional status of terms).

The next challenge for the project is to extend CPDs beyond syllogisms to cover predicate logic in full.

8 References

1. Cheng, P.C.-H.: Truth diagrams: An overview. In: B. Plimmer & P. Cox (Eds.), Proceedings of the 7th International Conference on the Theory and Application of Diagrams: Springer (2012, in press)
2. Cheng, P.C.-H.: Truth diagrams: A notation for propositional logic (in preparation)
3. Cheng, P.C-H.: Electrifying diagrams for learning: principles for effective representational systems. Cognitive Science, 26(6), 685-736 (2002)
4. Cheng, P.C-H.: Probably good diagrams for learning: Representational epistemic recodification of probability theory Topics in Cognitive Science 3(3), 475-498 (2011)
5. Cheng, P.C.-H.: Algebra Diagrams: A HANDi Introduction. In: B. Plimmer & P. Cox (Eds.), Proceedings of the 7th International Conference on the Theory and Application of Diagrams: Springer (2012, in press)
6. Cheng, P.C-H., & Barone, R.: Representing complex problems: A representational epistemic approach. In: D. H. Jonassen (Ed.), Learning to solve complex scientific problems. (pp. 97-130). Mahmah, N.J.: Lawrence Erlbaum Associates.
7. Copi, I.M., Cohen, C.: Introduction to Logic. Upper Saddle River, NJ: Prentice-Hall (1998)
8. Edwards, A.W.F.: Cogwheels of the mind: the story of Venn Diagrams. Baltimore, MD: John Hopkins University Press (2004)
9. Lemmon, E.J.: Beginning Logic. Wokingham, UK: Van Nostrand Reinhold (1965)

Understanding and Predicting the Affordances of Visual Logics

Jim Burton[1] and Peter Coppin[2]

[1] University of Brighton, UK
j.burton@brighton.ac.uk,
[2] University of Toronto, Canada
petercoppin@gmail.com

Abstract. We compare the affordances of two visual logics, one from the Euler family of notations, spider diagrams, and one which takes a significantly different approach to representing logical concepts, existential graphs. We identify strengths and weaknesses of each notation and present these features as being related to the idea that each notation is, to a greater or lesser degree, biased towards *objects* or *predicates*, and that such biases make a notation more or less effective in a given context. We then introduce a framework for understanding and predicting those affordances, which can help guide us towards better use of existing graphical notations and the design of more effective new notations. The framework links research in semiotics and linguistics with insights provided by the HCI and diagrams communities.

1 Introduction

A fundamental premise of the diagrams community is that graphical notations have, by some set of metrics which is not always made entirely clear, certain advantages over symbolic notations. These advantages relate to intuitive understanding and to the ability for new information to arise spontaneously within diagrams. Gurr [10] wrote that the effectiveness of a graphical notation arises from its being "well matched to meaning", which is to say that the syntax of the notation is naturally connected to its semantics. Hammer and Shin [11] showed that Euler diagrams do possess these advantages, and that while the changes made to Euler's notation by Venn and Peirce remove ambiguity and increase formal expressiveness, they also reduce its visual clarity.

If these advantages exist, and can be categorised and measured, the potential exists to design more effective graphical notations and to make better use of existing ones. Shin [23] undertook the latter task in her reevaluation of Peirce's system of existential graphs, in which she argued that if the diagrammatic properties of existential graphs were better understood and exploited in the design of reading procedures and inference rules, then they would be considered more useful as a tool for reasoning. Indeed, and in contrast to Euler diagrams, existential graphs have often been considered a cumbersome and non-intuitive system. In the same work, Shin shows, however, that Peirce consciously designed his system

3rd International Workshop on Euler Diagrams, July 2, 2012, Canterbury, UK.

to take advantage of distinctively diagrammatic properties, but that his insights were largely ignored in the way existential graphs were subsequently understood.

In this paper we will compare the affordances of two visual logics, spider diagrams [13] and existential graphs, analysing some of the strengths and weaknesses of each system. In this context, we use the term *affordance* to refer to the possible meanings of a piece of diagrammatic syntax, as perceived by an actor. The starting point for our comparison is the observation, made by Blackwell and Green [2], that "every notation highlights some kinds of information, at the cost of obscuring other kinds." We focus on the affordances arising from the spatial conditions of diagrams from each system with the same meanings. Thus, we consider the static properties of the notations and how those properties support comprehension, rather than any dynamic properties exhibited when either notation is used as a reasoning system. This work is a precursor to a planned empirical study in which we will test our findings. Our goal is not to show that one notation is superior to the other. In fact, existential graphs are considerably more expressive than spider diagrams. To enable the comparison, we will consider the fragment of existential graphs with monadic predicates only, which is equivalent to the spider diagram system. We choose the two systems for the comparison because we take them to be representative of two distinct families of visual logics: those based on Euler diagrams, such as spider diagrams and constraint diagrams [24], and logical graphs, such as existential graphs and conceptual graphs [5]. Spider diagrams and existential graphs are concerned with the same domain and have common features. For instance, both represent existential quantification directly. Neither system has an explicit way of representing universal quantification but both can do so implicitly. However, the two systems take fundamentally different approaches to representing information.

In sections 2 and 3 we examine the notational strengths and weaknesses of spider diagrams and existential graphs. We do so informally, introducing only so much of each notation as is necessary to make our argument. In section 4 we introduce Coppin's framework for visual affordances and show that it can be used to explain and predict the affordances described in the previous sections. The framework exposes general principles which we believe can be used to design effective visual notations, formal or otherwise. We show that the framework synthesises understandings gained from the fields of semiotics, neurolinguistics and diagrammatic reasoning. In section 5 we discuss the predictive power of the framework and the ways in which the principles of the framework may enable us to make more effective use of existing systems, such as spider diagrams and existential graphs, by understanding and exploiting their strengths.

2 Spider diagrams

Spider diagrams (SD) were introduced by Howse et al. in 2001 [13]. They are a sound and complete visual logic, equivalent in expressiveness to monadic first-order logic with equality [25]. Figure 1 shows a spider diagram, consisting of labelled *curves*, *spiders* and *shading*. Curves represent sets and their placement makes assertions about relations between sets: figure 1 tells us that the sets *Bird*, *Plane* and *Sman* are disjoint. Spiders are trees placed in the diagram, where the nodes are called *feet* and the edges are called *legs*. Each spider represents a single element that exists in one of the regions in which its feet is placed. The diagram in figure 1 includes a single spider, telling us that there is something which is either a bird, a plane, or Superman. Shading is used to represent the emptiness of regions. So, the shading in figure 1, considered alongside the information provided by the spider, tells us that *Sman* contains either one element or no elements.

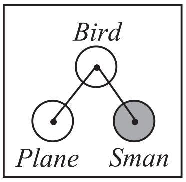


Fig. 1. Is it a bird, is it a plane...?

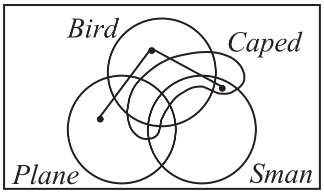


Fig. 2. A cluttered spider diagram.

The spider diagrams in figures 1 and 2 are *unitary* diagrams. SD allows us to use conjunction and disjunction to join together unitary diagrams to form *compound* diagrams. This is done by placing the usual symbols from symbolic logic in part of the diagram: see Howse et al. [13] for details.

As well as statements about sets, we can construct the meaning of a spider diagram as a series of logical assertions. From this point of view the diagram in figure 1 states, among other things, that $\exists x\,(Bird(x) \vee Plane(x) \vee Sman(x))$, and $\forall x\,(Bird(x) \Rightarrow \neg Plane(x))$.

SD extends the Euler diagram notation which, as noted in the previous section has been identified as being intuitive or fit-for-purpose as the basis of diagrammatic reasoning by several authors (see for, instance, chapter 6 of Shin [22], where the discussion focuses on Venn diagrams but also considers Euler diagrams). In summary, Euler diagrams represent relations between sets – intersection, disjointness, and so on – in a way that users can read quickly and intuitively because the spatial conditions “resemble”, in some sense, the properties they represent. Although a circle does not have any literal resemblance to the abstract notion of a set, a circle encloses a region of space and any point is either inside or outside of that region, just as any object is either inside or outside of a set. Thus, placing two circles so that they overlap or are disjoint leads the viewer to the obvious inferences about the relationship between the sets in question. Similarly, the fact that SD represents the existence of an element in a set by placing a spider foot in the region of the diagram that represents that set is well matched to meaning, and allows for intuitive understanding. Hammer and Shin [11] noted that the additions to Euler’s original notation do not always provoke the same natural associations in the reader. Shading, for instance, first introduced by Venn, bears no resemblance to the emptiness of a set and has a purely conventional meaning (apart from a slightly tenuous connection between shading, darkness and the emptiness of a void). Similarly, although spiders with a single foot may be well matched to meaning, the meaning of spiders with several feet is purely conventional. Thus, our first approach to understanding the intuitive power of diagrams might be to consider a spectrum from resemblance to conventionality. However, Shin [22] argues that resemblance is not, in fact, inversely proportional to conventionality. Two cognitive properties of diagrams which are inversely proportionate to each other, however, are conventionality and the use of *perceptual inferences*. That is, the less a notation relies on convention, the more perceptual inferences are introduced. Furthermore, she points out that several of the conditions we might want to represent are incapable of depiction without convention, particularly disjunctive and negative information. It is not possible to depict a situation that resembles the ones described by the formulae $A \vee B$ and $\neg A$. So, any graphical notation that conveys this type of information must do so by importing symbolic features. In SD, disjunction is shown using the symbolic device of spiders' legs and, in the full system, the logical symbol $\vee$. SD can depict some negative information by resemblance but not all. For instance, we can depict the situation reflected by the formula $\exists x(\neg A(x))$ by placing a spider outside of a curve labelled A, but to show $\neg\exists x(A(x))$, we must use the symbolic device of shading.

As well as inheriting the benefits of Euler diagrams, SD inherits some limitations: an Euler diagram can quickly become cluttered [15]. The diagram in figure 2 shows all possible intersections of four curves. We can see that this diagram has lost some of the readability of simpler examples, and the problem escalates quickly with the addition of more curves. If we want to add a new curve, say A, to figure 2 without adding any new information, we must do so such that A intersects every region, resulting in a diagram which is very difficult to draw and understand. The same problem applies to symbolic logic, however. A sentence from first-order logic that contains four or more predicate symbols or, worse, four or more variables, could also take considerable effort to read. Thus, SD, and visual logics generally, are not alone in suffering from clutter. There are ways of reducing this clutter, but these means, such as the use of discontinuous curves or overlapping edges [26], do so at the cost of some of the intuitive properties of Euler diagrams.

As a final observation about the diagram in figure 2, we note that we could produce this diagram from 1 by discarding three pieces of information about the disjointness of sets, as well as making other changes. Discarding this information requires us to *add* syntax to the diagram (the region representing $Bird \cap Plane$), and makes the resulting diagram harder to read as a result.

In summary, spider diagrams, at least in the case of unitary diagrams, preserve and extend much of the effective and intuitive power of the underlying Euler notation. However, the rapid accumulation of regions in a diagram can mean that SD doesn't scale well, although this lack of scalability also affects symbolic languages.

3 Existential graphs

Existential graphs were introduced by Peirce [17] at the end of the 19th century, at the same time as his seminal work on symbolic systems. There are two variations of the notation, α graphs and the more expressive β graphs, which are as expressive as first-order logic with equality [19]. Unlike SD, β graphs can represent predicates with any number of places. In order to make a meaningful comparison between the systems, we will consider Peirce's β graphs with the restriction that all predicates are monadic, and call this system EG. An graph in EG is composed of *predicate symbols*, *lines of identity* (LIs) and *cuts*. A predicate symbol is a label representing a predicate; since our predicates are monadic we can equally well consider the labels to represent sets as predicates. An LI is a network of lines which may have any number of branches and which represents an individual (there is an exception to this rule, which we explain below). A cut is a closed curve drawn on the diagram which represents the negation of that information inside it. The final syntactic device is *juxtaposition*: placing graphs G_1 and G_2 next to each other creates a new graph whose meaning is the conjunction of the meanings of G_1 and G_2. Figure 3 shows an existential graph. The parts of the graph labelled G_1 to G_5 we call *subgraphs* (note that these labels are added for convenience and are not part of the notation). The subgraph G_1 has one LI, three predicate labels and four cuts.

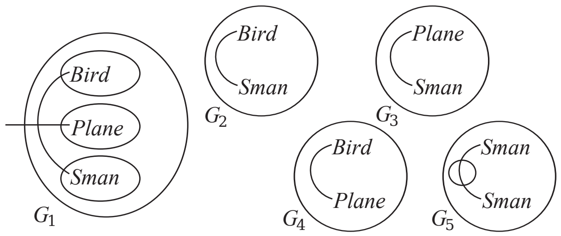


Fig. 3. An existential graph equivalent to figure 1

Interpreting existential graphs has often been seen as a difficult task, and this is one of the main points of criticism of the system. Shin [23] proposed a new reading procedure which is both more regular than earlier procedures and which exploits the diagrammatic properties of existential graphs. However, the reading procedure which is arguably easiest to describe informally is the *endoporeutic* reading proposed by Peirce himself. Informally, we read a graph from the "outside in", or from the region of the graph which is least enclosed by cuts towards the most enclosed part. Thus, reading subgraph G_2 in figure 3, we first encounter a cut, so we know that some piece of information is to be negated. Next, we encounter an LI, so we know that a statement concerning some individual will be made. Finally, the predicate labels *Bird* and *Sman* are at the ends of the LI, and we construct the meaning "it is not the case that there is some thing for which *Bird* and *Sman* are true", or "nothing is a bird and a Superman". More formally, we construct the formula $\neg\exists x(\mathit{Bird}(x) \wedge \mathit{Sman}(x))$. So, the subgraph G_2 in figure 3 conveys a subset of the information given in figure 1 by the placement of the curves labelled *Bird* and *Sman*. The conjunction of the meanings of the subgraphs G_2, G_3 and G_4 provides the same information as the placement of curves in figure 1. Reading G_1 from the outside in, we encounter an LI, denoting an individual, say x, so the constructed meaning begins with the fragment $\exists x \ldots$. Next, we encounter a cut, and so we have $\exists x \neg(\ldots)$. Next, we encounter three nested cuts, and we construct the fragment

$$\exists x \neg(\neg(\ldots) \wedge \neg(\ldots) \wedge \neg(\ldots)).$$

Inside these cuts are predicate labels attached to the ends of the LI, and we have

$$\exists x \neg(\neg(\mathit{Bird}(x)) \wedge \neg(\mathit{Plane}(x)) \wedge \neg(\mathit{Sman}(x))).$$

Shin's reading gives the equivalent but neater formula $\exists x(\mathit{Bird}(x) \vee \mathit{Plane}(x) \vee \mathit{Sman}(x))$. Thus, subgraph G_1, figure 3, demonstrates how disjunction is conveyed in EG. Compare this to figure 1, where the spider conveys the same information.

Subgraph G_5, figure 3, tells us that it is not the case that there are two things, say x and y, for which *Sman* is true and where $x \neq y$. That is, there is at most one Superman. This is the same information as is conveyed in figure 1 by the combination of the shading and spider foot in the curve labelled *Sman*. Inequality between objects in EG is shown by an LI which crosses an otherwise empty cut, indicating that the two (or more) extremities of the LI do not represent the same object; this is the exception to the way we read an LI mentioned above.

Thus, the spider diagram in figure 1 expresses the same meaning as the existential graph in figure 3. We note that figure 1 is much more compact than figure 3, demonstrating the expressive power of Euler diagrams and showing that, in this case, SD preserves and extends that power. Recall that spider diagrams assert information by the use of spiders, shading and the relative placement of curves. Figure 1 includes one spider, one shaded region and three curves placed so as not to intersect every region: five pieces of information. An existential graph representing the same information must include five subgraphs. Given a spider diagram, d, with n regions which are shaded or not represented in d, an existential graph, G, requires $n - 1$ subgraphs to represent the same information. The mapping between the spiders in d and subgraphs in G is not so straightforward, since several spiders which each have a single foot and which are placed in the same region can be represented by a single LI which crosses an other wise empty cut. The subgraphs of G will contain duplicated predicate labels, as is the case with figure 3, and distinct LIs which may refer to the same individual, whereas this duplication is not necessary in SD. This is one sense in which SD is more compact and elegant than EG. Like SD, the syntax of EG contains conventional or symbolic features: notably, cut bears no resemblance to negation. As discussed previously, however, any notation that represents negation or disjunction must do so symbolically. In comparison with Euler diagrams, it may seem that EG uses *only* symbolic features, with the exception of the LI. Even in this case, in which it seems reasonable to say that a line resembles, in some sense, an individual, the effect is marred by the special case of an LI that passes through an otherwise empty cut.

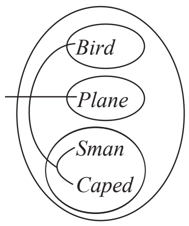


Fig. 4. An existential graph equivalent to figure 2

Similarly to the relationship between figures 1 and 3, the diagram in figure 4 has an equivalent meaning to the spider diagram in figure 2: informally, there is something which is either a bird, a plane, or is Superman and is wearing a cape. In this case, comparing the two diagrams leads us to the conclusion that EG is relatively effective in this context, since the intuitive properties of SD are hampered in figure 2 by clutter. In order to avoid asserting information about the relationship between two sets, S_1 and S_2, a spider diagram must include an unshaded region which contains no spiders and which represents $S_1 \cap S_2$. In figure 2, this leads to a diagram which is difficult to read (and draw). In EG, there is no need to avoid making claims about the relationship between S_1 and S_2. Adding more curves to the spider diagram would render it very difficult to read, whereas adding another predicate symbol to the existential graph in figure 4 would not have that effect. Furthermore, at the end of section 2 we noted that spider diagrams may require *more* syntax to represent *less* information. This is not the case with EG: the graph in figure 4 can be produced from the graph in figure 3 by adding one piece of information (that if a particular individual is Superman, it is also wearing a cape) and discarding three pieces of information regarding the relationships between the predicates (given by subgraphs G_2 to G_4 in figure 4). Discarding this information results in less syntax appearing in the graph in figure 4.

Remarkably, EG does not introduce specialised syntax to represent disjunction, conjunction or implication: all these properties are represented as a by-product of the spatial relations of predicate symbols, LIs, cuts and juxtaposition. Despite the fact that EG was almost entirely ignored by logicians until the 1960s, Peirce considered his graphical system superior to his own symbolic system [17]. Peirce categorised diagrammatic features as *icons*, *indices* and *symbols*. An icon represents something by its resemblance to that thing. An index represents something by “pointing it out”, much as a signpost does. A symbol represents some fact or condition merely by convention. Peirce’s aim was to create a system which was as iconic as possible. As discussed, this effort can be considered successful with respect to lines of identity, which “resemble” individual identity, but the use of cut is symbolic. Furthermore, the regularity of using LIs to represent individuals is disturbed by the special case of LIs which cross an empty cut, in which case a single LI represents two or more objects which are not the same. In contrast, the symbolic device of shading used in SD seems to us to pose less of a cognitive challenge, since the shading is deployed within the relatively iconic context of Euler circles.

Before describing Coppin’s framework in the next section, we note a final important syntactic similarity between SD and EG. Both systems feature node-link diagrams, which are lines of identity and spiders respectively. In EG the edges of the node-link diagrams represent identity and do so iconically, or by resemblance, while the nodes represent predicates, and do so symbolically. In SD the edges of the node-link diagrams represent disjunction and do so symbolically, whilst the nodes provide an iconic representation of individuals. In a recent eye tracking study [4], Burch et al. investigate the impact of the orientation and format of node-link diagrams on comprehension tasks; amongst other things, their results show that readers of node-link diagrams prioritise giving attention to *nodes* over *links* – these are the information-rich parts of the diagram. Thus, when comprehending the node-link component of a spider diagram, attention is first paid to a series of assertions about objects, making this an object-centric notation. When comprehending an existential graph, information about predicates is given prominence for the same reason, making EG predicate-centric. For our purposes, the relative efficiency of representation is of less importance than this bias towards objects (SD) or predicates (EG), accompanied by the fact that the nodal information is represented iconically by SD and symbolically by EG.

4 A framework for affordances

In this section we describe a framework that explains the fact that the differing approaches of SD and EG may each be more effective than the other in a given context.[3]

4.1 Pictorial and Symbolized Information

Our aim is to work towards an understanding of perception-recognition that can be used to distinguish pictorial and symbolized information. Throughout this development, it will be important to remember that perception of visual representations necessarily and simultaneously always involves both pictorial and symbolized information to various degrees. At the core of the argument is the

[3] The majority of the ideas in this section are attributed to author Coppin and will form part of his forthcoming PhD thesis [6].

claim that these two types of information closely correspond to two aspects of perception-recognition that we categorise as *emulation* and *simulation*. Indeed, it is via their relative engagements of these two aspects that we will be able to distinguish between symbolic and pictorial visual representations: pictorial representations engage relatively more of the aspects or perception that we characterize as emulation, while symbolic representations (that is, representations which contain relatively more symbolic information) engage relatively more of the aspects of perception we characterize as simulation. In order to proceed, we need to establish what is involved in these two key aspects of perception.

What we refer to as *emulation* can loosely be described as the aspect of perception-recognition that is most closely coupled with the proximal stimuli and sensations that impinge upon an organism. With respect to vision, emulation would include the near isomorphic response of retinal receptors to the aspects of the optic array [9] to which they are specifically tuned to respond. As information gets further from this "surface interface" and is processed by higher level aspects of the perceptual-cognitive system, it becomes less accurate to characterize the process as emulation. The key characteristic of this aspect of perception-recognition is that there is a structural relationship between the organism's response and the proximal stimulus (change) to which that response is a reaction.

What we refer to as *simulation* is alluded to by various terms in the cognitive science literature, such as"filling in" [18] and "prediction" [3]. This is the aspect of perception that allows us to see distal things as three-dimensional objects, even when only some subset of two-dimensional surfaces are reflecting light to our eyes. In order to achieve this, our visual systems must be able to simulate things and events in the world, in some spatio-temporal sense. This has been shown to rely on experience/memory and learning [12]. Because of this, the range of possibilities for a simulation that is a response or reaction to an external change or variation is greater than for the emulated aspects. Unlike emulation, because structural correspondence (between the proximal stimuli/change and the reaction) is not a defining characteristic of simulation, it is not easy or even possible to directly map back from the reaction to the structure of the stimuli. Emerging from all of this, the key characteristic of simulation is (subjective) extrapolation from the proximal structure of stimuli to the recognition of the distal structure of the world.

As described above, the distinction between these complementary modes of perception-recognition is noted at several points in the literature, and the modes are given various names. Hurford [14] identifies the modes with dorsal and ventral neural pathways, respectively. He notes the precedence of the dorsal/emulated mode and argues for the importance of this on the sense-making activities of our pre-linguistic ancestors and, ultimately, on our own development of language. Dorsal pathways make an initial categorisation of a stimulus that Hurford likens to a series of evaluations of predicate statements, whereas ventral pathways supply further environmental detail used to locate the stimulus in context. Thus, we use emulation to learn *what* a stimulus represents, before using simulation to learn *where* it is, how it stands in relation to its environmental context, and so on. To borrow terms from philosophy, we discover *quiddity*, the *whatness* or initial categorisation of a stimulus, then *haecceity*, the *thisness* or refined categorisation[4].

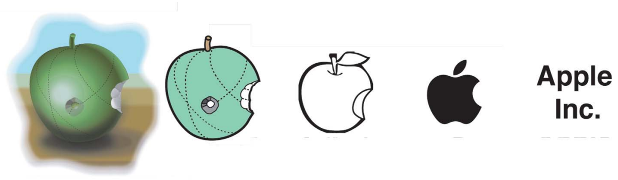
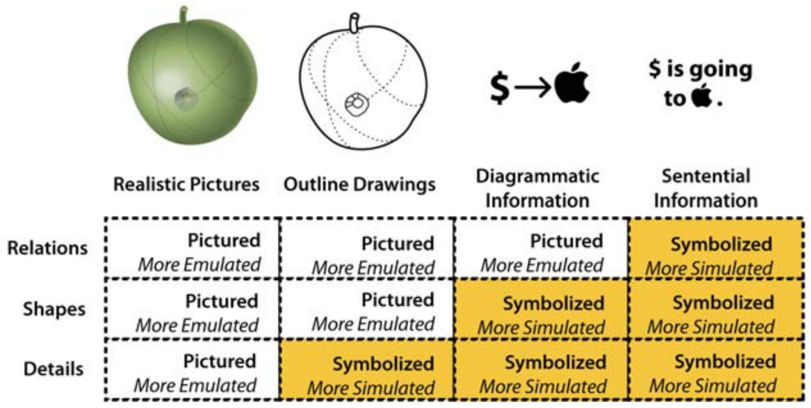


Fig. 5. From pictorial to symbolic representation.

At this point, an example is required. Figure 5 shows, from left to right, a realistic picture of an apple, successively less realistic depictions, the logo of the Apple company, and finally the (purely symbolic) name of that company. Emulation occurs when the viewer recognises that the apple picture on the far left is structurally similar to the emulation that occurs when looking at a real apple. Meanwhile, the activity that occurs when the viewer sees the Apple logo is less easily mapped back onto proximal stimuli and therefore more in the realm of simulation. Furthermore, the type and degree of learning required for perceiving-recognizing the Apple logo is greater than and different in quality from that required for perceiving-recognizing a photograph of an Apple. Together these two perceptual-cognitive distinctions justify distinguishing between the two representations such that we label one as being more pictorial and the other as being more symbolic.

Figure 6 presents these distinctions in grid form, and shows the mixture of emulation and simulation required to process content in a heterogeneous notation. To relate this to earlier sections, both SD and EG may be considered heterogeneous systems from this point of view, containing relatively pictorial and relatively symbolic elements. A good example of this is the differing semantics assigned to closed curves in each notation. In SD, the use of curves is strongly pictorial/emulated, whilst in EG it is strongly symbolic/emulated. Conversely, disjunction is symbolic/emulated in both notations. In particular, the node-link diagrams that feature in each notation are heterogeneous, and each notation mixes pictorial and symbolic information in opposite ways. In SD, nodes, which are spiders' feet, use the pictorial device of placement within a region, whilst edges, spiders' legs, are symbolic. In EG, nodes, which are predicate labels, are purely symbolic, whilst edges, lines of identity, are relatively pictorial.

[4] In Coppin's thesis the roles of memory and recognition are developed extensively in this context and these processes are posited as intermediaries between the two modes.



	Realistic Pictures	Outline Drawings	Diagrammatic Information	Sentential Information
Relations	Pictured *More Emulated*	Pictured *More Emulated*	Pictured *More Emulated*	Symbolized *More Simulated*
Shapes	Pictured *More Emulated*	Pictured *More Emulated*	Symbolized *More Simulated*	Symbolized *More Simulated*
Details	Pictured *More Emulated*	Symbolized *More Simulated*	Symbolized *More Simulated*	Symbolized *More Simulated*

Fig. 6. A perceptual-cognitive framework for graphic representation.

The Emulative and simulative modes are engaged, therefore, when processing node-link diagrams in either notation. Because priority is given to the nodes of those diagrams, however, the order in which the cognitive modes are engaged differs.

4.2 Affordances of Graphic Representation Types

The framework enables predictions regarding the perceptual-cognitive affordances of the graphic representation types described in this paper. We build on the idea that capabilities for emulation and simulation share a common, and limited, resource: attention and working memory [1, 8, 16]. Presented at a high level, the predictions we make are as follows.

1. Pictured object relations or attributes *interfere* with, or hinder, mental simulation of object relations or attributes, intended by an author.
2. Pictured object relations or attributes can *support* a mental emulation intended by an author.
3. Combinations of pictured and symbolized information can
   (a) free resources for mental simulation of symbolized objects, and
   (b) symbolized information affords mental simulations that are difficult, or impossible, to emulate.

We will first consider item 1. The "free rides" provided by Euler diagrams (and by other notations), noted by several authors and named by Shimojima [21], occur when information arises in a diagram as a by-product of its syntax. In figure 7, the Euler diagram on the left tells us that all jets are planes, and no birds are planes. We can read immediately from this, as a free ride, that no jets are birds. Free rides accumulate: we can produce the diagram on the right of figure 7 by adding the curve labelled *Sings* to the diagram on the left; as well as adding the information that everything that sings is a bird we also learn, as a free ride, that no planes can sing and neither can any jets sing.

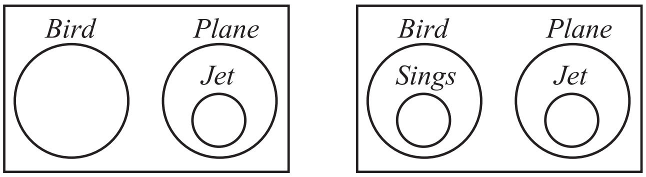


Fig. 7. Pictorial representations depict an entire state of affairs.

Free rides are a powerful component of the effectiveness of a graphical notation. They come about when a notation is well matched to meaning, allowing the viewer to use their intuition to make valid inferences for themselves. They also depend on a situation where a single piece of relatively pictorial syntax depicts *several things at once*, and this can become a significant distraction when comprehending a cluttered diagram. This point is related to the authors' previous work on constraint diagrams [7], in which we argue that generalized constraint diagrams possess the advantage over the original constraint diagram notation of being *less diffuse*; constructing the meaning of an individual piece of syntax can be done with reference to relatively fewer diagrammatic elements. However, the features that make generalized constraint diagrams less diffuse may also reduce the number of free rides available.

To consider cases in the current context where free rides may be counterproductive, recall that figures 2 and 4 display equivalent information. In figure 2, a relatively pictorial device is used to depict sets but since every possible set intersection is depicted, no information about the sets is conveyed by the curves in isolation. The informational content of the diagram relates to upper and lower bounds on set cardinality, and includes disjunctive information. This in formation is provided by shading and a spider, although the interpretation of the information depends on the relative position of curves. In the context of Coppin's framework, processing this information requires, predominately, simulation, and the framework predicts that a less pictorial approach may be effective. This is stated as item 1 above: *pictured object relations or attributes* interfere *with mental simulation.* In figure 4, we saw that EG conveys the same information as SD, figure 2, and we claimed that it did so relatively effectively. The framework predicts this effect, since in figure 4 the information requiring simulation is conveyed by predominately symbolic means.

On the other hand, consider figures 1 and 3. In figure 1, several pieces of information about relations between sets are conveyed simultaneously: no birds are planes, no birds are superman, and so on. Although this spider diagram also includes disjunctive information, the majority of the content is conveyed via the spatial relations of curves, and the viewer benefits from a natural mapping from these relations to relations between sets. The process of comprehending figure 1 is relatively emulative. We can see that a pictorial approach is effective by considering an existential graph with equivalent meaning to figure 1, shown in figure 3. In this figure no free rides occur and so each set relation is given explicitly using a symbolic representation. This is an example of the effect we state in item 2 above: *pictured object relations or attributes can* support *emulation.*

Figure 8 shows a spider diagram in which "uncertainty", or disjunctive information, is removed from figure 1 whilst figure 9 shows an existential graph with an equivalent meaning. The information conveyed by figures 8 and 9 is emulated, consisting of a series of initial categorisations: there are sets of birds, planes and supermen, there is a superman. Comparing the two representations shows that the pictorial approach of SD is certainly less cluttered and, we believe, more effective.

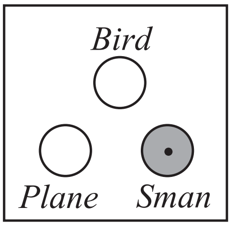


Fig. 8. It's Superman!

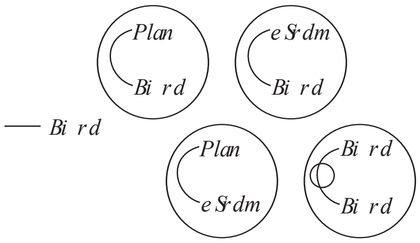


Fig. 9. Symbolic representation of emulated content.

Items 3a and 3b in our list of assertions on page 57 can be seen as corollaries to items 1 and 2. As we have seen, although the curves used in SD (and in Euler diagrams) to represent sets or predicates are more pictorial than the predicate labels of EG, the use of curves can reduce the effectiveness of the notation by causing clutter. This calls to mind Peirce's stated goal for EG ([17], quoted in Shin [23]) that a diagram should be "as iconic as possible": perhaps we should add to this the caveat "but not more". When something cannot be depicted, an approach that represents that information using a relatively symbolic device may be easier to comprehend and more scalable than one that uses a metaphor of resemblance.

5 Conclusion

Our comparison of the affordances of SD and EG is not undertaken in order to conclude which system is the most effective. Instead, we have shown that the interaction of pictorial and symbolic features can promote or hinder certain cognitive processes, which we call *emulation* and *simulation.* Using the framework, we have explained the fact there are certain tasks for which EG is surprisingly effective, although EG is (we believe) more cumbersome and less intuitive than SD. We have considered static comprehension tasks only, but SD and EG are reasoning systems. In further work, we intend to use the framework to evaluate the two notations when used to construct and comprehend proofs, and to conduct empirical studies which test the validity of the findings.

A more finely grained version of the predictions in section 4.2 is part of Coppin's thesis. Using these predictions in a consideration of emulated and simulated features in existing notations could lead to a principled approach to generating effective diagrams. The same problem is addressed, though using quite different means to our own, by Rodgers et al. [20] in their definition of well-formedness criteria for Euler diagrams and the effect of the criteria on readability. The fine grained principles could also be used by designers of new notations, through a consideration of the informational domain of the notation and the cognitive processes implied.

We also believe the framework can be used to investigate "layers" of information within graphical notations. As we have discussed, both SD and EG include node-link diagrams, and we believe the balance of pictorial/symbolic information at the nodes of these diagrams must be appropriate to the task in question. We conjecture that the node-link diagrams form part of an upper layer or "cognitive foreground" of the notations. Both SD and EG have a series of nested curves as a "background" layer, though only SD has a background layer which can be interpreted independently of other diagrammatic content. We intend to study the existence of layers of content and the effect of their degrees of independent coherence by conducting eye tracking studies that investigate the ways in which users pay attention to the syntactic elements of diagrams that include node-link diagrams amongst other syntax.

Acknowledgements: The authors would like to thank Steve Hockema and Gem Stapleton for helpful discussions around the topics in this paper.

References

1. A. Baddeley. Working memory. *Comptes Rendus de l'Académie des Sciences-Series III-Sciences de la Vie*, 321(2-3):167–173, 1998.
2. A. Blackwell and T. R. Green. Notational systems – the cognitive dimensions of notations framework. In John M. Carroll, editor, *HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science*, Interactive Technologies, chapter 5, pages 103+. Morgan Kaufmann, San Francisco, CA, USA, 2003.
3. A. Bubic, D. Y. Von Cramon, and R. I. Schubotz. Prediction, cognition and the brain. *Frontiers in human neuroscience*, 4, 2010.
4. M. Burch, N. Konevtsova, J. Heinrich, M. Hoeferlin, and D. Weiskopf. Evaluation of traditional, orthogonal, and radial tree diagrams by an eye tracking study. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2440–2448, December 2011.
5. M. Chein and M. Mugnier. Conceptual graphs: fundamental notions. *Revue dé Intelligence Artificielle*, 6:365–406, 1992.

6. P. Coppin. *Perceptual-cognitive properties of realistic pictures, outline drawings, diagrams, and sentences: toward a science of information design.* PhD thesis, University of Toronto, Available in 2012.
7. P. Coppin, J. Burton, and S. Hockema. An attention based theory to explore affordances of textual and diagrammatic proofs. In Ashok Goel, Mateja Jamnik, and N. Narayanan, editors, *Diagrammatic Representation and Inference*, volume 6170 of *Lecture Notes in Computer Science*, chapter 27, pages 271–278–278. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2010.
8. N. Cowan. An embedded-processes model of working memory. *Models of working memory: Mechanisms of active maintenance and executive control*, pages 62–101, 1999.
9. J. J. Gibson. *The ecological approach to visual perception.* Lawrence Erlbaum, 1986.
10. C. Gurr and K. Tourlas. Towards the principled design of software engineering diagrams. In *Proceedings of 22nd International Conference on Software Engineering*, pages 509–518. ACM Press, 2000.
11. E. Hammer and S. J. Shin. Euler's visual logic. *History and Philosophy of Logic*, pages 1–29, 1998.
12. S. A. Hockema. *Perception as prediction.* PhD thesis, Indiana University, 2004.
13. J. Howse, F. Molina, J. Taylor, S. Kent, and J. Gil. Spider diagrams: A diagrammatic reasoning system. *Journal of Visual Languages and Computing*, 12(3):299–324, June 2001.
14. J. R. Hurford. The neural basis of Predicate-Argument structure. *Behavioral and Brain Sciences*, 23(6), 2003.
15. C. John, A. Fish, J. Howse, and J. Taylor. Exploring the notion of clutter in Euler diagrams. In *4th International Conference on the Theory and Application of Diagrams*, pages 267–282, Stanford, USA, 2006. Springer.
16. G. A. Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956.
17. C. S. Peirce. *Collected Papers*, volume 4. Harvard University Press, 1933.
18. L. Pessoa, E. Thompson, A. Noë, and Others. Finding out about filling-in: A guide to perceptual completion for visual science and the philosophy of perception. *Behavioral and Brain Sciences*, 21(6):723–748, 1998.
19. D. D. Roberts. *The Existential Graphs of Charles S. Peirce.* Mouton, 1973.
20. P. Rodgers, L. Zhang, G. Stapleton, and A. Fish. Embedding wellformed euler diagrams. In *Thirteenth International Conference on Information Visualization*, pages 585–593. IEEE, 2008.
21. A. Shimojima. Inferential and expressive capacities of graphical representations: Survey and some generalizations. In *3rd International Conference on the Theory and Application of Diagrams*, volume 2980 of *LNAI*, pages 18–21, Cambridge, UK, 2004. Springer.
22. S. J. Shin. *The Logical Status of Diagrams.* CUP, 1994.
23. S. J. Shin. *The Iconic Logic of Peirce's Graphs.* Bradford Book, 2002.
24. G. Stapleton, J. Howse, and J. Taylor. A decidable constraint diagram reasoning system. *Journal of Logic and Computation*, 15(6):975–1008, December 2005.
25. 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.
26. G. Stapleton, L. Zhang, J. Howse, and P. Rodgers. Drawing euler diagrams with circles: The theory of piercings. *IEEE transactions on visualization and computer graphics*, September 2010.

The Online Abstraction Problem for Euler Diagrams

Gennaro Cordasco[1], Rosario De Chiara[2], and Andrew Fish[3]

[1] Dipartimento di Psicologia – Seconda Università di Napoli, ITALY,
gennaro.cordasco@unina2.it
[2] ISISLab, Dipartimento di Informatica – Università di Salerno, ITALY
dechiara@dia.unisa.it
[3] School of Computing, Engineering and Mathematics – University of Brighton, UK
Andrew.Fish@brighton.ac.uk

Abstract. A Euler diagrams are an accessible and effective visualisation of data involving simple set-theoretic relationships. Efficient algorithms to quickly compute the abstract regions of an Euler diagram upon curve addition and removal have been developed, but a strict set of drawing conventions (called wellformedness conditions) were enforced, meaning that some abstract diagrams are not representable as concrete diagrams. We present a variation and extension of the methodology which enables region computations for Euler diagrams under the relaxation of several drawing conventions. We provide complexity analysis and compare with the previous methodology. The algorithms are presented for generic curves, allowing for specialisations such as utilising fixed geometric shapes for curves that often occur in applications.

1 Introduction

Venn [21] and Euler diagrams are a well known representation of sets and their relationships. Venn diagrams have had significant theoretical interest from the likes of Grünbaum and Hamburger in recent times; a detailed survey of Venn diagrams can be found in [16]. Euler diagrams are the modern incarnation of Euler circles [3], first introduced for the purposes of syllogistic reasoning. Whilst Venn diagrams ensure that every region determined by being inside some contours and outside the other contours is nonempty, Euler diagrams generalise Venn diagrams by relaxing this condition. This allows them to specify subset relations and disjointness relations amongst sets without any extra cognitive load since these semantic relationships are well-matched the spatial relationships of containment and disjointness [11, 17].

In a practical setting, Euler diagrams appear frequently in various application domains. For example, they have been used in biological areas for representing complex genetic set relations in [14], in computer-based resource management scenarios in [2], and in the information retrieval/visualisation context to depict the numbers of results of collections of library database query results in [20] and in network visualisation [15]. Euler diagrams, together with diagrammatic inference rules, form a diagrammatic logic, and comparisons of the effect of the choice of inference rules on automated searches for minimal proofs within Euler diagram-based reasoning systems [18] has been investigated. There are many variations of the basic system, and they have also been incorporated into heterogeneous reasoning systems [19]. More complex diagrammatic logics such as Spider [12] or Constraint diagrams [4, 13] build on the underlying Euler diagram logic, adding more syntax in order to increase the expressiveness of the languages.

3rd International Workshop on Euler Diagrams, July 2, 2012, Canterbury, UK.

Motivation. For any computer-based applications there is a natural disparity between the concrete level information that the user perceives and manipulates (the drawn or concrete diagrams) and the abstract information that the system requires or manipulates (the abstract models or abstract diagrams). Many important computations of the system tend to be defined at this abstract level. For instance, if one wished to present the semantics of a user-constructed diagram then the system needs to perform computations such as to identify the regions present in the diagram, to compute the set intersections that they represent, and to combine these into some set-theoretic statement. In a more general sense an efficient way to calculate the abstract diagram is also useful for the comparison of concrete diagrams.

The efficient computation of the abstract model from a given concrete diagram, together with the ability to update the abstract model upon concrete changes such as curve addition, removal, translation and resizing represent an important challenge to be addressed. The relaxation of the drawing constraints is a significant extension because under these relaxed constraints, every abstract diagram has a concrete diagram representing it [5]. Furthermore, for dynamic diagrams (e.g. sequences of diagrams constructed during interactive user construction or the presentation of evolving data sets) it permits the temporary relaxation of the chosen set of drawing conventions imposed for a diagram in the sequence. This enables a natural construction or presentation, assisting in preserving a user's mental map.

Contribution and paper outline. In this paper we provide a solution to the *on-line abstraction problem*: compute the abstraction of a concrete Euler diagram (i.e. a drawn diagram), keep track of the concrete and abstract diagrams, and enable the automatic update of the abstract diagram upon concrete level manipulations. The algorithms presented in [8] solved this problem for the wellformed diagrams of [5], but here we provide a solution for the more general case in which several well-formedness conditions are relaxed, enabling much greater utility and flexibility. The algorithms have also been extended to permit further relaxation of the wellformedness constraints, enabling the processing of 'generalised Euler diagrams' (representing sets as unions of regions with holes), ensuring that any abstract diagram has a concrete realisation. However, this further extension is not presented in this paper due to space constraints.

2 Preliminaries

We provide a definition of Euler diagrams, separating the abstract and concrete models as usual, together with the set of wellformedness conditions considered. Specifically, we incorporate some of the wellformedness conditions of [5] and [10] that are simplicity of contours (no self-intersection) and uniqueness of contour labels, into the main definition of concrete Euler diagram, which enables an omission of labels since the contours can be uniquely identified.

Definition 1. *A* concrete Euler diagram *is a pair $d = \langle \mathcal{C}, \mathcal{Z} \rangle$ where $\mathcal{C}$ is a set of simple closed curves, called* (concrete) contours*, in the plane and $\mathcal{Z}$ is the collection of* (concrete) zones *z determined by being inside a set of contours $X_z \subseteq \mathcal{C}$ and outside the rest of the contours. That is,*

$$z = \bigcap_{c \in X_z} int\,(c) \cap \bigcap_{c \in \mathcal{C} - X_z} ext\,(c)\,,$$

for each $X_z \subseteq \mathcal{C}$, provided this region is non-empty. Here $int(c)$ and $ext(c)$ denote the interior and the exterior of c, respectively, and the set X_z is called the zone descriptor *for z. A* minimal region *of d is a connected component of* $\mathbb{R}^2 - \bigcup_{c \in \mathcal{C}} c$.

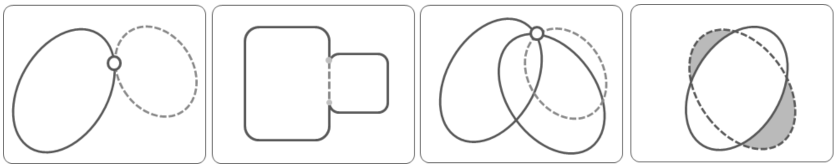


Fig. 1. Nonwellformed Euler diagrams, breaking WF1a, 1b, 2 and 3, resp., from left to right.

We say d is wellformed *if the following wellformedness conditions (WFCs) hold (see Fig. 1):*

WF1 ***Transverse intersections****: Contours that intersect do so transversely.*
This can be subdivided into:
WF1a *No tangential intersections.*
WF1b *No concurrency (distinct contours meet at a discrete set of points).*
WF2 ***No multiple points****: At most two contours can intersect at any given point.*
WF3 ***Connected concrete zones****: Each concrete zone is a* minimal region.

An abstract Euler diagram (see Definition 2) is an abstraction (see Definition 3) of a concrete diagram. We overload the term zone, using it for the concrete zones, which are regions of the plane, as well as for abstract zones, which are the sets of containing contours of that region (or the labels of those contours in the generalised case); the context determines which is meant. Let $\mathcal{P}X$ denote the powerset of set X.

Definition 2. *An* abstract Euler diagram *is a pair: $d = \langle C(d), Z(d) \rangle$ where: $C(d)$ is a finite set of labels, called* (abstract) contours*, drawn from some alphabet $\mathcal{L}$. The set of* (abstract) zones *of d is $Z(d) \subseteq \mathcal{P}C(d)$, where $\bigcup_{z \in Z(d)} z = C(d)$.*

Definition 3. *Let d be a concrete Euler diagram and d' an abstract Euler diagram. If there is a bijection between $\mathcal{C}(d)$ and $C(d')$ that induces a bijection between $\mathcal{Z}(d)$ and $Z(d')$, then d is said to be a* realisation *of d′, and d′ is the* abstraction *of d. An abstract Euler diagram d′ is* drawable *if there is a realisation of d′ as a concrete Euler diagram.*

By convention, each concrete Euler diagram contains a zone o, called the *outer zone*, which is exterior to all the contours (that is, $X_o = \emptyset$). The left of Fig. 2 shows a concrete Euler diagram containing four contours (A, B, C, D). The zone descriptors for the concrete zones are graphically depicted in the right hand side of the figure; these sets can be viewed as the abstract zone set.

We need terminology relating to the important operations of the addition and removal of the contours of an Euler diagram.

Definition 4. *Let $d = \langle \mathcal{C}, \mathcal{Z} \rangle$ be a concrete Euler diagram with $A \notin \mathcal{C}$ and $B \in \mathcal{C}$. Let $d + A$ and $d - B$ denote the concrete Euler diagrams obtained by the addition of a new*

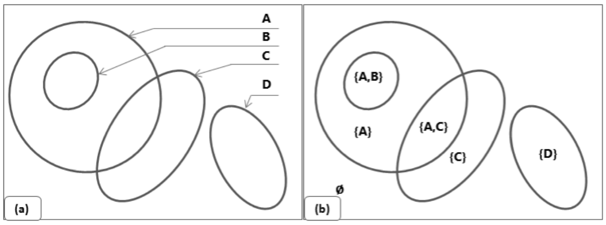


Fig. 2. (a) A wellformed concrete Euler diagram, with contour identifiers (or labels); (b) a depiction of the zone descriptors.

contour A to d and the removal of contour B from d, respectively. A region r of d is a union of minimal regions; it is: (i) a covered region (or is covered *by A) if $r \subset int(A)$ in $d + A$; (ii) split by A (a split region) if $r \cap int(A) \neq \emptyset$ and $r \cap ext(A) \neq \emptyset$ (i.e. r is partially covered by A). Analogously, a zone z of d is a* covered zone *(respectively a split zone) when it is covered (respectively partially covered) by A.*

Fig. 3 shows an example of contour addition. We observe that the zone described by $\{C\}$ is split by the contour A but neither of its two minimal regions is split by A. A point of intersection x between two curves c_1 and c_2 is called a *crossing point*. We denote with $Cross(d)$ the set of all the crossing points generated by contours in d.

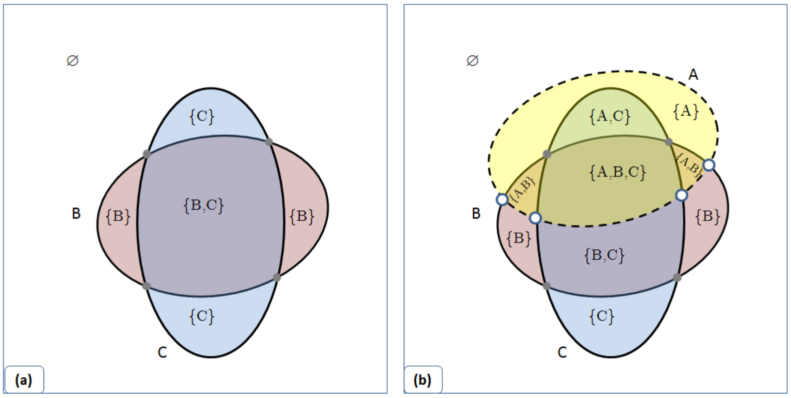


Fig. 3. An example of contour addition: (a) A non wellformed diagram $d = \langle\{B, C\}, \{\emptyset, \{B\}, \{C\}, \{B, C\}\}\rangle$. The crossing points of d are shown with filled-in dots; (b) The crossing points of A with d (i.e. those in $Cross(A)$) are depicted as hollow dots. The set of all hollow and filled-in dots depicts the set of crossing points of $d + A$.

3 Computing the abstraction of Euler Diagrams

The main problem that we address is the following, with variations according to the choice of wellformedness conditions imposed.

Abstraction Update Problem

Let $d = \langle\mathcal{C}, \mathcal{Z}\rangle$ be a concrete Euler diagram and $d' = \langle C', Z'\rangle$ the abstraction of d. Let $A \notin \mathcal{C}$ and $B \in \mathcal{C}$. Efficiently compute the abstractions of $d + A$ and $d - B$.

In [6, 8] the *single marked point approach* (SMPA) (described below) was presented, computing the abstraction for wellformed Euler diagrams, following an online approach where diagrams are viewed as a sequence of contour additions and removals. Other operations such as translation or resizing of a contour can be easily simulated by the addition and removal operations, and so such algorithms are applicable in a wider context.

In this paper, we present an evolution of these algorithms, adopting the *multiple marked point approach* (MMPA) (described below), permitting the relaxation of condition WF3 so that Euler diagrams whose zones are disconnected can be processed.

First of all, we recall from [8] that the set of zones split by A, due to the addition of a contour A to (or its removal from) a given wellformed Euler diagram d, can be computed using the following observation.

Observation 1 *Let d be a wellformed Euler diagram and let $\{x_0, x_1, \ldots, x_{m-1}\}$ be all of the crossing points that we meet as we traverse the contour A from an arbitrary point on A. Then:*
(i) For each $i = 0, \ldots, m-1$ each arc $(x_i, x_{i+1 \bmod m})$ splits one zone (note that two arcs can split the same zone but one arc cannot split more than one zone) of d.
(ii) Two consecutive arcs $(x_i, x_{i+1 \bmod m})$ and $(x_{i+1 \bmod m}, x_{i+2 \bmod m})$ split two zones such that their zone descriptors differ by exactly one contour $($the contour which intersects with A generating the crossing point $x_{i+1 \bmod m})$.

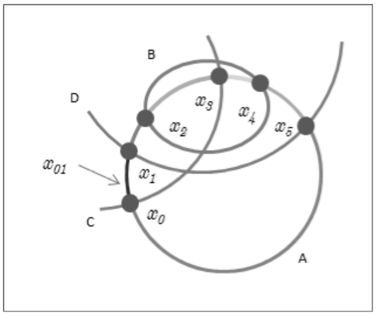


Fig. 4. A schematic diagram which illustrates Observation 1. The addition of A generates six new crossing points shown with small blobs. The point x_{01} is an arbitrary point of the arc (x_0, x_1) used to compute an initial zone descriptor for the zone split by arc (x_0, x_1), whilst subsequent zone descriptors are computed using Observation 1.

For the wellformed diagram case of [8], we adopted the SMPA where each zone z of d has a single point $mp(z) \in \mathbb{R}^2$ associated to it, where $mp(z)$ lay in the boundary of the closure of the zone z (with the possible exception of the marker for the outside zone). These points keep track of the zone sets, and were used to update these sets according to their relationships with contours that are added or removed from a diagram. Then, the zones that are not split by A are covered by A if and only if $mp(z)$ belongs to the interior of A. The SMPA is illustrated in Figure 5: each minimal region is marked by a single marked point (an arrowed dot indicates a marked point, the arrow indicating the minimal region which is marked); additional marked points, or *pseudo-crossing points*, are used to mark the outside zone and any contour which has no crossing points, either in d or at some stage during its incremental construction (e.g. see the marked point for zone $\{B, D, E\}$ in Figure 5). However, the SMPA is not sufficient to deal with the Euler diagrams with disconnected zones. In this case, there are two ways of splitting a zone: (i) a zone is split when one of its minimal regions is split by A; (ii) a zone is split when some of its constituent minimal regions are covered by A and some are not. To address case (ii) one can consider associating one marked point to each minimal region of the diagram. Figures 5 (c) illustrates a generalization of the case of [8] where each minimal region is associated with one marked point. Then, if a zone z has no minimal regions which are split by A (i.e. case (i) does not hold) we can analyse the relationships of the marked points with A to classify z as split by A, covered by A, or neither. In particular, if all of the marked points of z belong to $int(A)$ then z is a covered zone, whilst if some of the marked points of z belong to $int(A)$ while others do not, then z is a split zone, according to (ii) above.

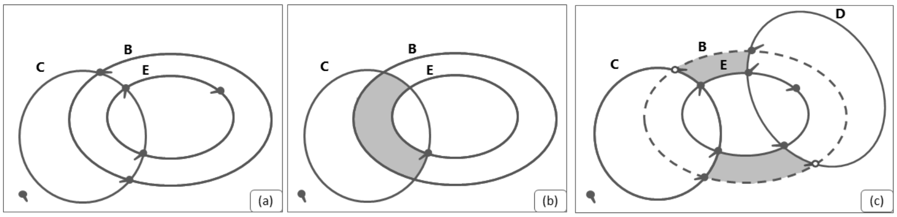


Fig. 5. Single marked point approach (SMPA): in (a) each zone of a wellformed Euler diagram is marked by a single point; (b) shows in grey the zone $\{B, C\}$ and its marked point; in (c) a non wellformed Euler diagram (WF3 relaxed) with a zone $\{B\}$, shown in grey, which consists of two minimal regions, therefore requiring at least two marked points.

However, the management of marked points (taking one for each minimal region) for non-wellformed diagrams (relaxing WF3) raises some tricky problems such as: if a zone z becomes split upon contour addition or deletion, how can one efficiently find a marked point for each of the minimal regions that comprise z? For example, Figure 6 shows two parallel examples which adopt the SMPA (using the algorithm of [8]) in which only the order of contour addiction has been varied. Whilst one of these gives a valid solution, the other does not. In detail, the first two steps (a) and (b) in the figure represent the addition of the first two contours (E and C) to the diagram. Then the two cases are depicted: on the left we add first contour D and then contour B, whilst on the right we first add B and then D. In the first case (on the left) the association between the marked points and minimal regions is correct, with the two minimal regions of zone $\{B\}$ marked by points z_0 and z_2. However, in the second case (on the right) the association is incorrect: there are two marked points associated to the same minimal region (top, shaded) and no marked point associated the other minimal region (bottom, shaded). The problem is that, in general, there is no easy way of discriminating between the case on the left from the case on the right; by just analyzing the relation between a marked point and the contours it is possible to discriminate between zones and not minimal regions.

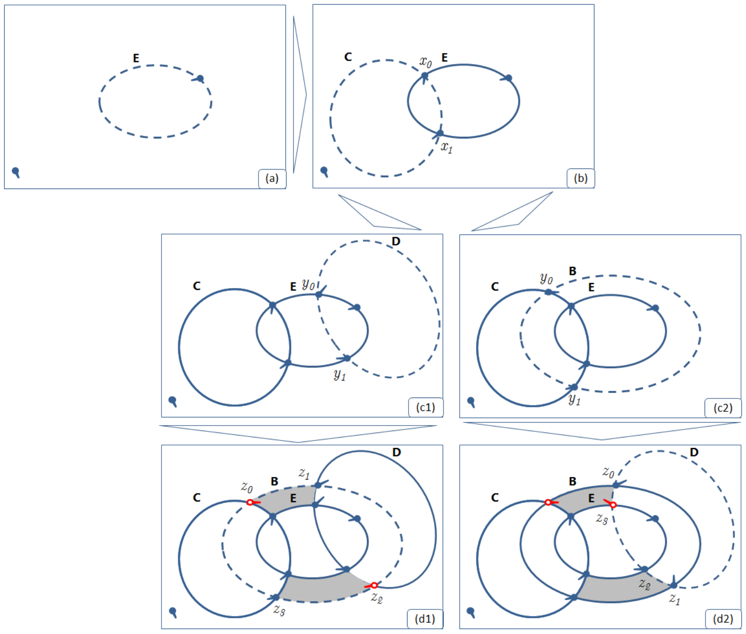


Fig. 6. The influence of the order of contour addition on the marked points/minimal region association, by using the algorithms of [8]. The dotted contour is the one that is going to be added to the current diagram.

We avoid such problems by adopting the MMPA in which each zone is associated with a set of marked points (which comprises the set of crossing points laying on its boundary). This approach requires the tracking of a larger number of marked points but we accept this trade-off against a simpler marked point management (also making implementation easier), since when a zone is split, we just need to correctly partition the set of its marked points.

The MMPA is illustrated in Figure 7: a set of points marks each minimal region r, including all of the crossing points on the boundary of r. Thus, each zone has marked point set including all of the crossing points laying on its boundary (i.e the boundaries of its constituent minimal regions). In the specific case in Figure 7, each marked point marks one, two or four zones.

In the following we define a procedure for contour addition (with contour removal omitted for space reasons), which satisfies:

Theorem 1. *Let $d = \langle \mathcal{C}, \mathcal{Z} \rangle$ be an Euler diagram, satisfying WF1 and WF2 (i.e. with WF3 relaxed). Then*

i) *If $A \notin \mathcal{C}$, then the procedure **NewContour**(d, A) computes the new collection of zone descriptors for the zones $\mathcal{Z}'$ of $d' = \langle \mathcal{C} \cup A, \mathcal{Z}' \rangle$.*

ii) *If $B \in \mathcal{C}$, then the procedure **DeleteContour**(d, B) computes the new collection of zone descriptors for the zones $\mathcal{Z}'$ of $d' = \langle \mathcal{C} - B, \mathcal{Z}' \rangle$.*

Moreover, both procedures:

1. *compute, for each zone $z \in \mathcal{Z}' - \{z_0\}$, where z_0 is the zone in the exterior of all contours in d', the set of marked points of the zone, $MP(z)$, that is comprised of the*

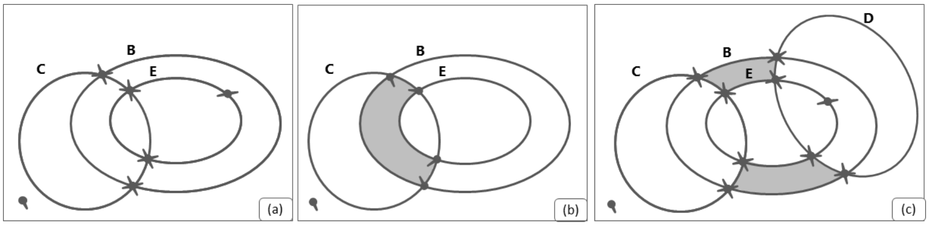


Fig. 7. Multiple marked point approach (MMPA): in (a) each zone is marked by points including all of the crossing points belonging to its boundary; (b) shows in grey the zone $\{B, C\}$ and its marked points; (c) a non wellformed Euler diagram (WF3 relaxed) with a zone $\{B\}$, shown in grey, comprised of two minimal regions, utilising eight marked points.

set of all crossing points (or pseudo-crossing points) of d' belonging to the closure of z. There is a single marked point $mp(z_0)$ in the exterior of all of the curves of d'.

2. *have running time $O(|\mathcal{Z}| + |Cross(d)| \log(|Cross(d)|))$.*

3.1 The algorithms

Initially, we consider the key case of Euler diagrams with WF3 relaxed (but WF2 and WF1 enforced). Subsequently, we will provide the ideas enabling the relaxation of WF2 and WF1. For space reasons we present only the algorithm for contour addition (omitting the algorithm for contour removal). Both algorithms are based on two auxiliary algorithms, **ComputeContourRelationships** (which computes the relationship of a contour with the other contours in a diagram, and updates the marked point sets) and **ComputeSplitRegions** (which computes the zone descriptors of the split zones using Observation 1).

Definition 5. *Let $d = \langle \mathcal{C}, \mathcal{Z} \rangle$ be a concrete Euler diagram and A a contour which is not in $\mathcal{C}$. Let*

$Over(A)$ *denote the collection of all of the contours in $\mathcal{C}$ that properly overlap A; that is $Over(A) = \{c \in \mathcal{C} \mid int(A) \cap int(c) \neq \emptyset\}$;*

$Cont(A)$ *denote the collection of all of the contours in $\mathcal{C}$ that properly contain A; that is $Cont(A) = \{c \in \mathcal{C} \mid c \notin Over(A)$ and $int(c) \cap int(A) = int(A)\}$.*

For example, Figure 3 (a) shows diagram d and Figure 3(b) shows the addition of contour A to d. We have $Over(A) = \{B, C\}$, $Cont(A) = \emptyset$ and $Cross(A)$ contains the four crossing points between A and the contours in $\mathcal{C}$.

The methodology adopted makes use of the following low level computations, and we assume that, given two contours A and B of an Euler diagram with WF3 relaxed, we can quickly find:

1. the relationships between A and B; that is if A and B properly overlap, or if one contains the other;
2. their crossing points (if A and B properly overlap);
3. the relationship between any given point $x \in \mathbb{R}^2$ and A; that is whether x belongs to $A, int(A)$ or $ext(A)$.

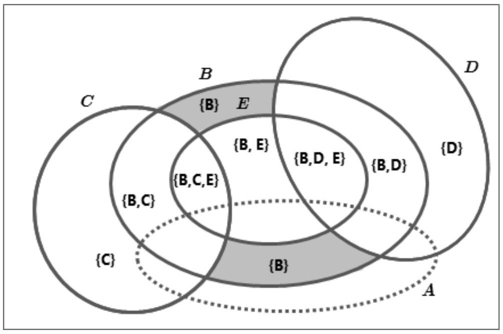


Fig. 8. The addition of contour A splits eight minimal regions determined by the eight arcs that comprise A. However, it splits nine zones, eight of which are the distinct zones containing the eight minimal regions that are split. The ninth zone $\{B\}$ is split, without splitting any of its constituent minimal regions, since one of its minimal regions is covered by A but the other is not.

Placing restrictions on the geometric shapes used for contours (which is common in some applications) can enable particularly fast computations. For example, if each contour is a simple geometric shape, such as a circle or an ellipse, these computations reduce to solving a system of two quadratic equations (1 and 2) and a quadratic equation (3), which can be computed very quickly (with different methods having different time/precision tradeoffs).

Algorithm **ComputeContourRelationships**: (i) computes the relationship between the contours present in a diagram d (with WF3 relaxed) and a contour A; (ii) updates the set of crossing points of d.

We will refer to Fig. 8 to assist with the explanation of the algorithms. Consider the addition of the dashed contour A to the diagram in Fig. 8 without A. After the execution of Algorithm **ComputeContourRelationships** we have $Cont(A) = \emptyset$, $Over(A) = \{B, C, D, E\}$ whilst $Cross(A)$ is the set of eight crossing points created by the addition of A.

Algorithm **ComputeSplitRegions** uses the sets output by Algorithm **ComputeContourRelationships** and calculates the collection of zone descriptors for the zones that contain a minimal region of $d - A$ split by A. The crossing points of A can be used to decompose A into a set of arcs. This algorithm computes the zone descriptors of all of the zones of $d - A$ that have at least one of their minimal regions split by A.

In detail, the arcs are analysed in the sequence that they are met as one traverses the contour; see Fig. 4 for an example. The region that is split by the first arc (x_0, x_1) is determined the set of contours that properly contain A and then by checking which of the contours that properly overlap with A contains the arc. Each successive region that is split is calculated by computing the difference with the previously computed region; this idea was present in Observation 1.

Contour addition. Algorithm 1 updates the collection of zone descriptors upon the addition of a new contour A to a diagram d. There are two cases to consider. Firstly, if $Over(A) = \emptyset$ then no new crossing points are created by the addition of A, and so A forms a new connected component. Thus, A splits only the zone described by contours

Algorithm 1: NewContour(d, A)

```
Input: An Euler diagram $d = \langle \mathcal{C}, \mathcal{Z} \rangle$ and a contour $A$ such that $A \notin \mathcal{C}$.
Output: $\mathcal{Z}'$, the collection of zone descriptors of $d' = \langle \mathcal{C} \cup \{A\}, \mathcal{Z}' \rangle$.
1:  $(Cont(A), Over(A), Cross(A)) :=$ ComputeContourRelationships$(d, A)$
2:  if $Over(A) = \emptyset$ then            // $A$ does not properly overlap any contour present in $\mathcal{C}$
3:      $s := Cont(A)$                                        // $s$ is the zone split by $A$
4:      $Z_s := \{s\}$                  // $Z_s$ is the set of zones having a minimal region split
5:      $s.points :=$ any point in $A$                          // the marked point for $s$
6:  else
7:      $Z_s :=$ ComputeSplitRegions$(d, A, Cont(A), Over(A), Cross(A))$
8:  $\mathcal{Z}' := \mathcal{Z}$
9:  forall $z \in Z_s$ do
10:     $s := z$                                                     // $s$ is the old zone
11:     $n := z \cup \{A\}$                                          // $n$ is the new zone
12:     $M_s := M_n := M_A := \emptyset$
13:     forall $x \in s.points$ do
14:         switch do
15:             case $x \in int(A)$                                 // the point $x$ marks $n$
16:                 $M_n := M_n \cup \{x\}$
17:             case $x \in ext(A)$                                 // the point $x$ marks $s$
18:                 $M_s := M_s \cup \{x\}$
19:             case $x \in A$                          // the point $x$ marks both $s$ and $n$
20:                 $M_A := M_A \cup \{x\}$
21:     $s.points := M_s \cup M_A$
22:     $n.points := M_n \cup M_A$
23:     $\mathcal{Z}' := \mathcal{Z}' \cup \{n\}$ // The new zone $n$ is added to the collection of zones of the diagram
24: forall $z \in \mathcal{Z} - Z_s$ do
25:     $M_{int} := M_{ext} := \emptyset$    // $M_{int}$ and $M_{ext}$ record the marked points of $z$ that are in the
        interior and the exterior of $A$, respectively
26:     forall $x \in z.points$ do
27:         if $x \in int(A)$ then
28:             $M_{int} := M_{int} \cup \{x\}$
29:         else
30:             $M_{ext} := M_{ext} \cup \{x\}$
31:     if $M_{ext} = \emptyset$ then              // if $z$ is covered by $A$ then $z$ should be removed
32:         $\mathcal{Z}' := \mathcal{Z}' - \{z\}$
33:     else
34:         $z.points := M_{ext}$
35:     if $M_{int} \neq \emptyset$ then    // if $z$ is split or covered by $A$ then a new region should be added
36:         $n := z \cup \{A\}$
37:         $n.points := M_{int}$
38:         $\mathcal{Z}' := \mathcal{Z}' \cup \{n\}$
39: return $\mathcal{Z}'$
```

in $Cont(A)$ (in Fig. 2 (a), contour D splits only the outer zone, exterior to all other contours, for example). Secondly, if $Over(A)$ is not empty, then A splits several zones (contour A in Fig. 8 splits several zones, for example). Algorithm 1 computes the split zones in two steps: (i) the split zones which contain a split minimal region are computed by Algorithm **ComputeSplitRegions**; (ii) the split zones which do not have any split minimal regions are computed, together with the set of covered zones, by analysing the relationship between the collection of marked points of the zones and the contour A.

For instance, in Fig. 8, the zones having a minimal region split by A are $\{\emptyset$, $\{C\}$, $\{B, C\}$, $\{B, C, E\}$, $\{B, E\}$, $\{B, D, E\}$, $\{B, D\}$, $\{D\}\}$ while the zone $\{B\}$ is split by A even though neither of its two minimal regions are split by A since one of them is covered by A and the other is not.

In detail, when $Over(A)$ is empty, A does not properly overlap any of the contours in d and it does not generate any new crossing points. In this case there is exactly one zone of d split, defined by $Cont(A)$, and any point on A can be chosen as the marked point for both of the zones of $d+A$ that are described by $Cont(A)$ and $Cont(A) \cup \{A\}$, referred to as the *old zone* and the *new zone* respectively (lines $3 - 5$).

Thereafter the algorithm considers the marked points of each zone of d which has a minimal region that is split by A (line 7). The variable s refers to the zone in d that is split as well as the old zone that this becomes upon the addition of A in d', the variable n refers to the new zone that is created in d' from s which is also inside A. For each such marked point x, the algorithm checks if x is a marked point for just the old zone, just the new zone or both (lines $13 - 20$) in d' (recorded using M_s, M_n and M_A respectively). The new zone n is added to the collection of zones of the diagram (line 23). Subsequently (line 24) the algorithm checks the remaining zones (i.e. those that do not contain any split minimal regions) looking for covered or split zones of d. This is performed by verifying the relationship between the marked points of the zone $z \in \mathcal{Z} - Z_s$ and A (lines $26 - 30$). In particular, if all of the marked points belong to $int(A)$ (i.e., $M_{ext} = \emptyset$) then the zone z is covered by A and so the old zone z is removed from the diagram (lines $31 - 32$). Moreover, if at least one marked point belongs to $int(A)$ then the zone z is either split or covered and so a new zone is generated and added to the diagram (lines $35 - 38$).

3.2 Relaxing the wellformedness condition WF2 and WF1

WF2. We extend the algorithms to also handle crossing points with multiplicity greater than 2 (i.e., more than two contours crossing transversely at a given point). For Euler diagrams with WF3 relaxed, we used Observation 1 in Section 3 to compute the set of zones split by the addition (or removal) of a contour A. Although Observation 1 (i) holds for Euler diagrams with WF2 also relaxed, part (ii) does not hold if the crossing point $x_{i+1 \bmod m}$ has multiplicity greater than 2. Fig. 9 (left) presents a schematic diagram, similar to that in Fig. 4, in which there is a crossing point, x_2, with multiplicity 3. Observation 2 provides the modification of the strategy to deal with diagrams that relax WF2 (as well as WF3).

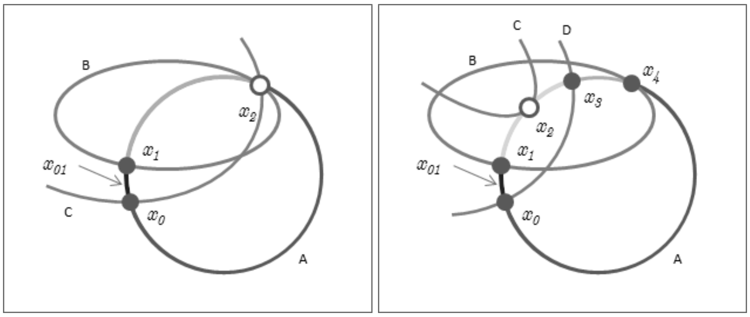


Fig. 9. (left) A schematic diagram illustrating the need for Observation 2 when WF2 is relaxed (c.f. Fig. 4). (right) The addition of A, yielding a diagram with WF1a relaxed. We have four transverse crossing points (shown as filled dots) and one tangential intersection point (shown as a hollow dot).

Observation 2 *Let d be an Euler diagram with WF3 and WF2 relaxed. Let $\{x_0, x_1, \ldots, x_{m-1}\}$ be all of the crossing points that we meet as we traverse the contour A from an arbitrary point on A. If there are exactly $\ell \geq 1$ contours crossing A transversely at a point $x_{i+1 \bmod m}$ then the zone descriptors of the zones that are split by the arcs $(x_i, x_{i+1 \bmod m})$ and $(x_{i+1 \bmod m}, x_{i+2 \bmod m})$ differ by exactly ℓ contours, and these are the contours that intersect with A comprising the crossing point $x_{i+1 \bmod m}$.*

The algorithms in Section 3.1 can now be altered to deal with the relaxation of WF2.

WF1. Firstly, we relax WF1a. Since tangential intersection points do not affect the zones which are split by the corresponding arc (see Fig. 9 (right)), we adapt the algorithm to deal with tangential intersections by simply ignoring them. Hence even the relaxion of WF1a is straightforward.

Secondly, we relax WF1b, allowing concurrency. For this case, we assume that, given two contours A and B of an Euler diagram, we can quickly: (i) check whether they meet in a concurrent arc (i.e. a maximal non-discrete set of points of intersection). (ii) check whether a concurrent arc is tangential (i.e. if a homotopy of the concurrent arc to a point would leave a tangential intersection point; see Figure 10 left) or transversal (i.e. if a homotopy of the concurrent arc to a point would leave a transverse crossing; see Figure 10 right) and (iii) find the split points (the points where two contours that meet in a concurrent arc separate).

The split points will play essentially the same role as the crossing points within the extended algorithms: they will be used as marking points for zones and to compute the set of zones which are split by the addition of a new contour A (noting that the crossing points are also still used, as before). There are two cases to consider: (i) tangential concurrent arcs and (ii) transversal concurrent arcs. For a tangential concurrent arc, both of the split points of that arc do not affect the zones which are split (see Figure 10 left), and so we treat such points in the same manner as tangential intersections; For a transversal concurrent arc, the first split point that we encounter as we traverse the contour A does not affect the zone which is split, whilst the corresponding second split point does affect the zone which is split (see Figure 10 right). Therefore, the first point will be treated as a tangential intersection while the second one will be treated as a transverse crossing.

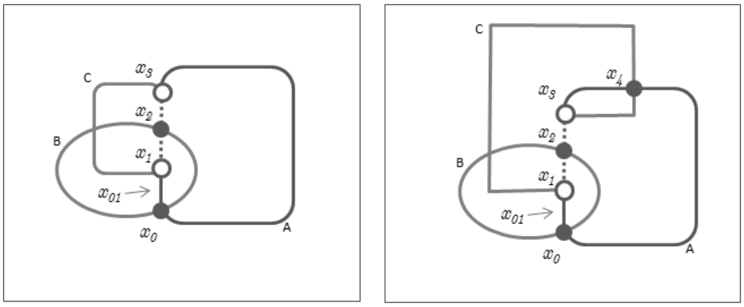


Fig. 10. The addition of A, yielding diagrams with WF1b relaxed. (left) The contour A creates two transverse crossing points, x_0 and x_2, and one tangential concurrent arc between points x_1 and x_3 (shown as hollow dots). (right) The addition of contour A create three transverse crossing points, x_0, x_2 and x_4, and one transverse concurrent arc between points x_1 and x_3 (shown as hollow dots).

3.3 Timing

We provide complexity analysis for the MMPA for Euler diagrams.

The invocation of procedure **ComputeContourRelationships** analyses the relationship between A and each contour in $\mathcal{C}$ and updates the set of crossing points, taking time $O(|\mathcal{C}| + |Cross(d)| \log(|Cross(d)|))$.

Then, if there are intersection points, the procedure **ComputeSplitRegions** computes the set Z_s of zones having a split region and, for each such zone updates the set of marked points. Hence the procedure **ComputeSplitRegions** requires $O(|\mathcal{C}| + |Cross(d)| \log(|Cross(d)|)$ steps.

Finally, for contour addition, lines $9-23$ and $24-38$ respectively compute the collection of marked points for zones having, and not having, split minimal regions, respectively. We can compute this two collections in $O(|\mathcal{Z}| + |Cross(d)|)$ steps. Collectively, algorithm **NewContour** operates within time $O(|\mathcal{Z}| + |Cross(d)| \log(|Cross(d)|))$.

4 Related work and conclusion

The relationship of the Euler diagram abstraction problem with arrangements of Jordan curves in the plane [9] was discussed in [6]. We note that in our approach we do not need to store or manipulate graphs, but we work directly with the diagrams utilising its intersection points and the domain specific data structures, and we obtain a methodology with a straightforward means of implementation.

In [22], an application is presented that interprets an Euler Diagram sketched with a pen or a mouse, and calculates the abstract diagram. The authors claim complexity that is asymptotically similar to ours, but this claim is not substantiated, with the paper not providing details of how the Zone List Refinement Step is performed. The only apparent method that would be effective with the generality that they describe is a pixel based inspection of the drawing (commonly available in programming languages) but which has the drawback of being dependent on the resolution of the image. Our methodology has the added advantage of being more general in that it is not dependent on the image of the contours, but only on their analytical representation.

In [23], a methodology is provided which takes a set of polygons (meaning regions determined by sets of non-overlapping curves) and outputs a set of non-overlapping polygons, which is essentially the boundary of a zone of the diagram in our terminology; this enables the computation of polygon operations such as union, intersection, difference and clipping. A graph based representation is constructed which consists of a binary tree structure, encapsulating the structure of non-overlapping contours, together with a winged-edge data structure which captures sets of polygons as a graph (indicating the intersection points) and provides a simple means of traversing faces of the graph. Their algorithm "corrects" input containing degeneracies (e.g. zero area contours or coincident edges, meaning concurrency), whereas we wish to develop a method that explicitly considers them. For 'diagrams' that consist of more than one connected component, they compare each output contour with the others to determine if one is inside the area of another or if they co-exist within the same area, and they record these contour relationships in the hierarchical tree structure of the contours. Our approach (utilising marked points) provides removes the need to compute these graph structures and then to operate on them, whilst providing a means to explicitly capture the 'singularities' that may occur within certain contexts or application domains.

In [1], they prove that the Grünbaum encoding uniquely identifies simple Venn diagrams (i.e. they are wellformed) which are monotone and polar symmetric, and develop an algorithm utilising a matrix representation to enumerate the monotone simple symmetric 7-Venn diagrams. The codes considered in the paper rely upon numbering the curves (adopting certain conventions based on the curve segment in other and inner faces to fix the choices) and for a given curve recording the sequence of curve numbers that are given as one traverses the curve. Our methodology applies to a much wider class of diagrams, but investigating the computation of encodings from the data structures utilised in our algorithms is an interesting line of future investigation.

In this paper, we have developed a new methodology, called the multiple marked point approach, which enabled us to develop algorithms to solve the Euler diagram abstraction problem for nonwellformed Euler diagrams relaxing the constraints imposed on the previous state of the art in [8], which utilised the single marked point approach and was limited to the wellformed diagram case. Furthermore, these algorithms extend in a natural manner to enable the processing of generalised Euler diagrams (i.e. unions of regions with holes).

This enables greater flexibility for users of software tools, enabling them to work with nonwellformed diagrams where it is convenient, or necessary, to do so. For example, during user construction of wellformed diagrams if the user is permitted to construct intermediary non-wellformed diagrams then this aids them in adopting a natural construction method, whilst it can be very complex, or even impossible, to do so without allowing passage through non wellformed diagrams. Furthermore, the extension to generalised Euler diagrams ensures that every abstract diagram has a concrete realisation (which is not the case for wellformed diagrams, as shown in [5]).

The algorithms presented in this paper are available in a Java library, although to simplify implementation (and to improve the efficiency) it restricts the geometric shape of contours to be arbitrarily rotated ellipses [7]. This enables the specification of each contour in parametric form, enabling easy implementation of operations such as choosing a point on an arc, or checking contour relationships of intersection or containment.

The applications of this work are widespread since the algorithms can be utilised in any software system that utilise Euler diagrams or their extensions.

References

1. T. Cao, K. Mamakani and F. Ruskey. Symmetric Monotone Venn Diagrams with Seven Curves. In *Fifth Intern. Conference on Fun with Algorithms*, LNCS 6099, 331-342, 2010.
2. R. De Chiara, U. Erra, and V. Scarano. VennFS: A Venn diagram file manager. In *Proceedings of Information Visualisation*, pages 120–126. IEEE Computer Society, 2003.
3. L. Euler. Lettres a une princesse dallemagne sur divers sujets de physique et de philosophie. *Letters*, 2:102–108, 1775. Berne, Socit Typographique.
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, A. Fish, and J. Howse. Euler diagram generation. *Journal of Visual Languages and Computing*, 19:675–694, 2008.
6. G. Cordasco, R. De Chiara, and A. Fish. Interactive Visual Classification with Euler Diagrams. In *Proceedings of VL/HCC*, pages 185–192. IEEE Press, 2009.
7. G. Cordasco, R. De Chiara, and A. Fish. EulerDiagramNWF: source code. http://isis.dia.unisa.it/projects/EulerNWF, 2010.
8. G. Cordasco, R. De Chiara, and A. Fish. Efficient on–line algorithms for Euler diagram region computation. *Comput. Geometry: Theory and Application (CGTA)*,44:52–68, 2011.
9. H. Edelsbrunner, L. Guibas, J. Pach, R. Pollack, R. Seidel and M. Sharir. Arrangements of curves in the plane—topology, combinatorics, and algorithms. In *Theoretical Computer Science*, Vol. 92 N. 2, pages 319–336. Elsevier Science Publishers Ltd. 1992.
10. A. Fish. Euler Diagram Transformations. *Graph Transformations & Visual Modelling Techniques, ECEASST*, 18:1–17, 2009.
11. C. A. Gurr. Effective Diagrammatic Communication: Syntactic, Semantic and Pragmatic Issues. *Journal of Visual Languages and Computing*, 10:317–342, 1999.
12. J. Howse, F. Molina, J. Taylor, S. Kent, and J. Gil. Spider Diagrams: A Diagrammatic Reasoning System. *Journal of Visual Languages and Computing*, 12(3):299–324, 2001.
13. S. Kent. Constraint Diagrams: Visualizing Invariants in Object Oriented Models. In *Proceedings of OOPSLA97*, pages 327–341. ACM Press, 1997.
14. H. Kestler, A. Muller, T. Gress, and M. Buchholz. Generalized Venn diagrams: A new method for visualizing complex genetic set relations. *J. of Bioinformatics*, 21(8), 2005.
15. N. Henry Riche, and T. Dwyer. Untangling Euler diagrams. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1090–1099, 2010.
16. F. Ruskey. A survey of Venn diagrams. *Electronic Journal of Combinatorics*. www.combinatorics.org/Surveys/ds5/VennEJC.html, 1997
17. A. Shimojima. Inferential and Expressive Capacities of Graphical Representations: Survey and Some Generalizations. In *Proceedings of 3rd International Conference on the Theory and Application of Diagrams*, Vol. 2980 of *LNAI*, pages 18–21. Springer-Verlag, 2004.
18. G. Stapleton, J. Masthoff, J. Flower, A. Fish, and J. Southern. Automated Theorem Proving in Euler Diagrams Systems. *Journal of Automated Reasoning*, 39:431–470, 2007.
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*, Vol. 134 of *ENTCS*, pages 189–202, 2005.
21. J. Venn. On the Diagrammatic and Mechanical Representation of Propositions and Reasonings. *Phil.Mag*, 1880.
22. M. Wang, B. Plimmer, P. Schmieder, G. Stapleton, P. Rodgers, and A. Delaney SketchSet: Creating Euler diagrams using pen or mouse. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing VL/HCC*, pages 75–82. IEEE Press, 2011.
23. K. Weiler. Polygon comparison using a graph representation. *Computer Graphics (SIGGRAPH '80 Proceedings)*, 14(3):10–18, July 1980.

Aesthetic and Practical Concerns in the Drawing of Euler and Venn Diagrams: Case Studies using SVG

David Dailey

Computer Science Department, Slippery Rock University, Slippery Rock, PA, USA
david.dailey@sru.edu

Abstract. SVG is a graphical language standardized by the World Wide Web Consortium for the display of vector graphics on the WWW. It would appear to be precisely appropriate for drawing Euler and Venn diagrams so that those drawings would scale properly to different display devices. This paper presents techniques for using SVG for doing exactly this as well as investigating a variety of aesthetic considerations that might improve the "explanatory power" of such diagrams. Such features as the smoothness and convexity of curves, the angles and distances between their intersections, the areas of resultant regions, colors and other markings of the curves, as well has how to bring about these effects are considered.

1 Introduction

SVG or Scalable Vector Graphics has been the World Wide Web Consortium's (W3C) recommended standard for web-based vector graphics since 2001 [1]. Its implementation by web browsers, however, has been somewhat slow, with Opera, Firefox, Safari and Chrome all providing some support during the years 2005 to 2008, but Microsoft not providing support, finally, until 2011, with the release of Internet Explorer 9 [2] .

SVG is an XML language that provides formulaic descriptions of graphics that allow quicker download times than bitmapped formats (such as GIF, JPEG and PNG), rescaling to the device resolution (so that clarity may be preserved as an image is zoomed in upon or printed), and interactivity (meaning that images may be modified dynamically through JavaScript). By being a W3C standard, SVG ensures interoperability with the other web standards such as HTML, CSS, DOM, JavaScript and AJAX.

It is a language particularly well-suited to the creation of mathematical illustrations such as graphs [3], Voronoi diagrams [4], and, as will be argued in this paper, Euler and Venn diagrams. In part, this advantage owes to the portability and scalability of this form of graphics (supported by all modern web browsers and about 1 billion cell phones [5]), but SVG also brings advantages toward addressing certain aesthetic concerns, as these diagrams are used as explanatory constructs in teaching disciplines ranging from Computer Science to Psychology and Philosophy [6] (`http://www.jfsowa.com/ontology/ontoshar.htm`). This paper will also present some of these aesthetic concerns and will critically examine SVG's suitability for addressing them.

Overall, the paper will conclude that while SVG is very appropriate and, in fact, easy to use to create illustrations that will be of value to the diagramming community, some of the more advanced features of the language (like filters and animation) still suffer from inconsistent support across browsers, while other aspects of the language appear to suffer from under-specification within the W3C specification. It is hoped that some of these concerns can be addressed within the nascent SVG 2.0 specification currently under development by the W3C.

2 Several Aesthetic Concerns for the Presentation of Set Theoretic Diagrams

Intrinsic to the study of Venn and Euler diagrams are several terminological distinctions and a few presuppositions that seem to be motivated by aesthetic considerations. For example, the concept of a *simple Venn diagram* [7] is based on the notion that no more than two curves should intersect at a given point. The very definition of Venn diagrams begins with the concept of the *simple closed curve*, despite the fact that certain set relations might be better represented by complex closed curves such as shown in figure 1. And included in the basic definition of Venn diagrams is another primarily aesthetic concern: the notion that all distinct regions in the plane should have differing inclusion relations (i.e., that no two subregions should be subregions of exactly the same collection of regions). We also, generally speaking, prefer smooth closed curves rather than ones with sharp edges, and prefer the regions each to be "large enough" (certainly to be visible).

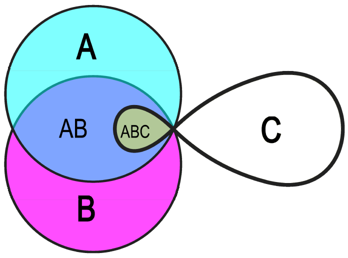


Fig. 1. An Euler diagram in which the absence of certain regions (AC and BC are null) motivates the use of closed curves that are not simple. This image is available in vector format at `http://cs.sru.edu/~ddailey/svg/NotSimple.svg`.

This section will discuss several aesthetic concerns that can be, in some sense of "perceptual distinguishability," be added naturally to these diagrams to enhance their "readability."

2.1 Aesthetic Criteria for Venn and Euler Diagrams

1. Curves should be, when possible, simple closed curves. The example of Figure 1 illustrates a situation in which, for sake of accurate adherence to the set relations, this may not be possible.
2. Curves should be smooth. Certainly the investigation of polygonal diagrams is of interest to combinatorists, but smooth curves are easier for the eye to follow.
3. When possible, curves should be convex. This appears not always to be possible, and the constructions of n-Venn diagrams, for any n, by both Venn and Edwards [7] rely on curves that have numerous concavities. But curves that do not have frequent changes of direction (from convex to concave) are intuitively easier for the human eye to follow.
4. The number of intersections between any pair of curves should be not only finite, as required by the standard definition, but small. Once again, this relates to the perceptual ease of understanding a diagram, though two completely convex curves may intersect at arbitrarily high numbers of points (as, for example, two regular n-gons each centered at the same point, with one slightly rotated about the center).
5. The number of curves that meet at a given point should be minimal. While many inquiries of Venn diagrams seek to minimize the number of intersections in the entire diagram (to make the graph of the diagram have fewer nodes), such diagrams are harder for the eye to traverse and must often rely on additional visual cues such as color to make the various curves distinguishable. According to this criterion, diagrams whose graphs are four-regular are preferred.
6. The points of intersection between regions should be relatively far apart, when possible. Cleary, if more than two curves intersect at a given point, then the distance has been eliminated altogether, but again, for ease of perception, it is best to ensure that these intersections are not so close together that confusion should result.
7. When two or more curves intersect at a given node, the angles of intersection should be "relatively high." For example, in the four diagrams in Figure 2, diagram 1, is the relatively "classical" presentation, with each circle passing through the center of the other, producing an angle of intersection of sixty degrees. In Figure 2, the angels between the curves is maximized at ninety degrees. While it is arguable that diagram 1 may be preferable to diagram 2, this would probably stem from the relative sizes of the regions, a factor addressed in criterion 8 which follows.
8. The regions of the Venn diagram should be relatively "large." While in certain contexts, we may wish to ensure that the areas of regions is proportional to the cardinality of the sets and intersections represented by the diagrams [8], absent this empirical constraint, it is expected that the human perceiver will more readily make sense of the subset and overlap relations when the regions are large enough to be labeled, shaded, or colored, and certainly, large enough to be seen.

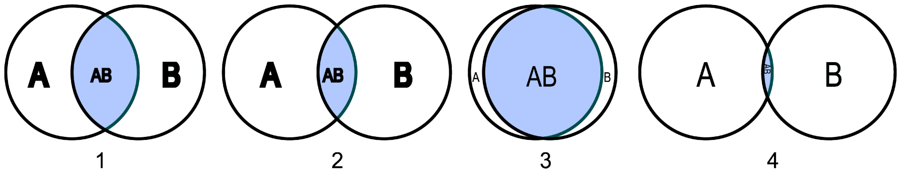


Fig. 2. Diagrams illustrating different sizes of regions and angles of intersection. This image is available in vector format at `http://cs.sru.edu/~ddailey/svg/intersection.svg`.

9. Regional markings (coloration, patterning or shading) should maintain "family traits." That is, each of the subregions of a curve should bear some resemblance to one another, and certainly to the "undiluted" pure set as it appears outside the intersections with other sets. In Figure 3, the first two diagrams (at upper left) present the traditional additive and subtractive color models as rendered in SVG. And though, each subregion *inherits* its chroma from its parents, this fact is rather lost on many human perceivers who see no particular resemblance between black or white and the surrounding colors.
10. Notwithstanding criterion, 9, the markings (coloration, patterning or shading) of regions should, nevertheless, be maximally distinguishable, from one another, particularly when regions are adjacent. In the sixth diagram of Figure 3 (lower right) the four lowermost regions within it are not so perceptually distinguishable as we might prefer.

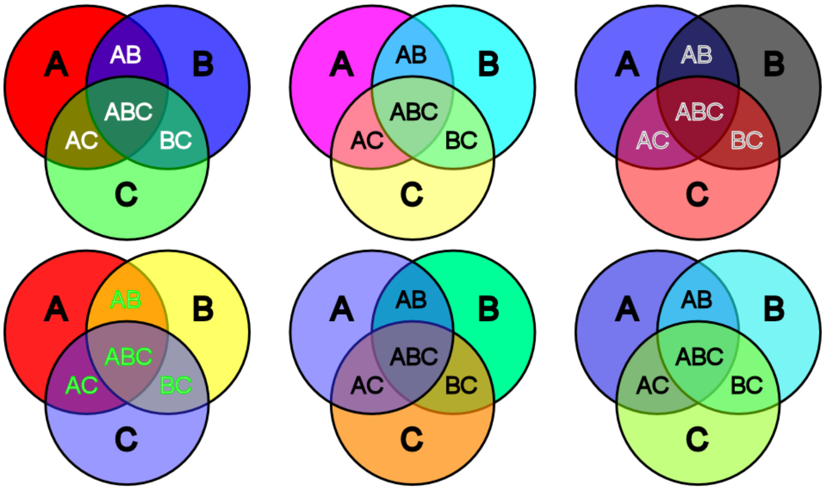


Fig. 3. Various Venn diagrams marked with different colors in an attempt to satisfy criteria 9 and 10. This image is available in vector format at `cs.sru.edu/~ddailey/svg/V11.svg`.

Among these ten criteria are many principles that rather work against one another. Distinguishability of regions (criterion 10) and family resemblance (criterion 9) call for opposing solutions. Likewise, as seen in Figure 2, maximizing the angles of intersection of curves, is in contrast with maximizing the areas of the regions inside, particularly for the preferred convex curves. This, however, is the nature of aesthetics: a compromise, ultimately between several competing criteria. One seeks to minimize confusability at several different parts of a rather complex psychophysical realm. In subsequent sections, we'll look at how some approaches to working in this realm using SVG.

3 Basics of Using SVG

Those readers familiar with HTML will find SVG quite natural. It is a markup language complete with tags (inside angle brackets "<" and ">") and attributes. The tags of SVG are typically geometric things like `<line>` `<circle>` `<rect>` (for rectangle), and the attributes are things like r=20 that might specify the radius of a circle, or width=200 that might specify the width of a rectangle. Sometimes, for sake of reusing code, we may group objects together using the group element `<g>`, and reuse them using the `<use>` element. SVG also has transforms (for rotation, translation and resizing), patterns (for specifying customized patterns for use in filling curves), and filters (for modifying the way that regions appear).

SVG is visible in all modern browsers (meaning that you will have to use Internet Explorer 9 or 10 to see it) and many mobile devices (including i-phones, Androids that run IceCreamSandwich, Blackberry Playbooks, and dozens of others) for a total number of devices estimated at 1 billion [5]. However the SVG specification [9] is complex and has taken several years for most browsers to implement. Currently, support is quite thorough and consistent for basic SVG, but is a bit spotty for more advanced features like filters and animation [10].

To get started, one may copy the following code (also available at `http://cs.sru.edu/~ddailey/svg/simplest.svg`) into a text editor and save it as *simplest.svg*. Once saved in a place that is visible to web browsers (typically, a web server, though one's own hard drive will generally work), then one may simply open the file with a (modern) browser to see.

```
<svg xmlns="http://www.w3.org/2000/svg">
<circle r="50"/>
</svg>
```

This should succeed in drawing a black circle centered at the extreme upper left corner of the web browser. To some extent, it is the simplest SVG document possible.

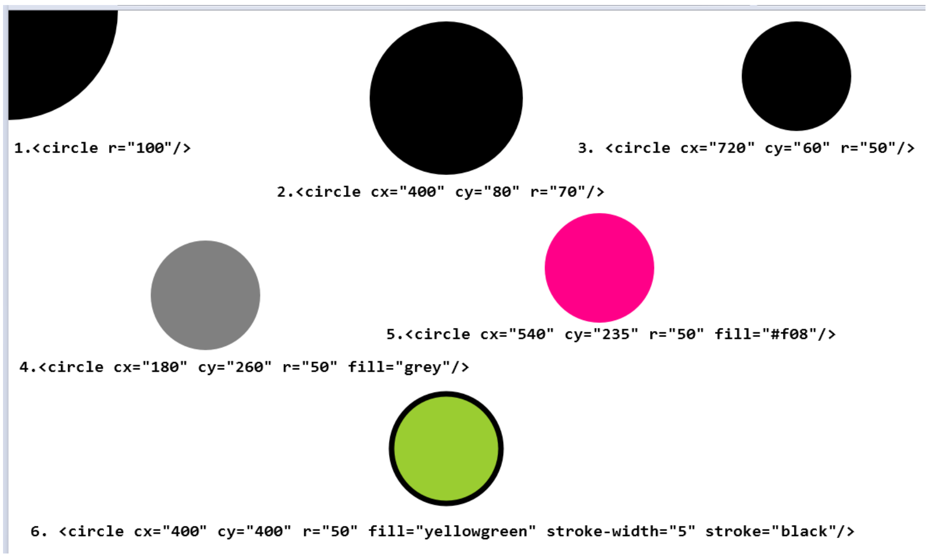
Much more about how to use SVG to draw far more elaborate illustrations, and even interactive web documents, can be seen in the W3C's SVG Primer [2].

4 Drawing Curves: using circles, ellipses and paths.

SVG has several drawing primitives, including polygons, rectangles, and the like, but for these purposes we will concentrate on how to draw smooth curves: circles, ellipses and paths.

4.1 Drawing circles

While the above *simplest* code adds only one attribute (r for radius) to the `<circle>`, more attributes may be added to vary its center, the colors of both its boundary and interior, and other aspects. As shown in Figure 4, the coordinate system places the origin (0,0) at the top left of the browser window, with x increasing to the right and y increasing downward.

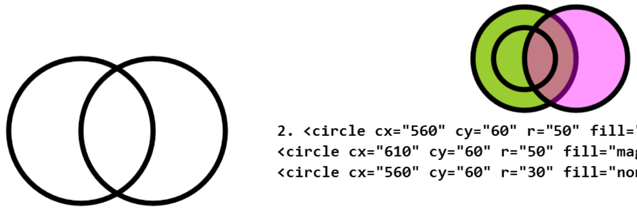


```
1.<circle r="100"/>
3. <circle cx="720" cy="60" r="50"/>
2.<circle cx="400" cy="80" r="70"/>
5.<circle cx="540" cy="235" r="50" fill="#f08"/>
4.<circle cx="180" cy="260" r="50" fill="grey"/>
6. <circle cx="400" cy="400" r="50" fill="yellowgreen" stroke-width="5" stroke="black"/>
```

Fig. 4. Circles drawn with increasingly more complexity and attributes. This image is available in vector format at `cs.sru.edu/~ddailey/svg/circles.svg` .

As can also be verified from the above, *fill* controls the color (or gradient or pattern) that is placed into a shape; its default value is *black*. If the center of the circle (cx,cy) is not specified, then it is assumed to be (0,0) .

Two other things about circles (and by extension the more complex shapes like ellipses and paths) in SVG should be known. We may make the interior of a circle invisible by setting fill=”none” and we may vary the opacity of the stroke and the fill, independently as shown in Figure 5. Both of these will prove useful in drawing of set theoretic diagrams.



```
2. <circle cx="560" cy="60" r="50" fill="yellowgreen"/>
<circle cx="610" cy="60" r="50" fill="magenta" fill-opacity=".4"/>
<circle cx="560" cy="60" r="30" fill="none"/>
```

```
1.<circle cx="130" cy="130" r="70" fill="none"/>
<circle cx="200" cy="130" r="70" fill="none"/>
```

Fig. 5. Making a circle empty or partly transparent. This image is available in vector format at `cs.sru.edu/~ddailey/svg/opacity.svg`.

The magenta circle in the second diagram of Figure 5 has its opacity set to .4. Hence it will be 40% opaque and 60% transparent. The stroke-opacity attribute has not been adjusted, meaning that its value remains the default: 1.0, or 100% .

4.2 Ellipses

Ellipses are drawn in SVG much like circles except for having two radii: rx, and ry representing the distances from the center to the horizontal and vertical extremities. And unlike circles, rotation is relevant for ellipses, allowing us, for example, to draw, Venn's symmetric construction of one of the two 4-Venn diagrams. This example shows some of SVG's power for creating these sorts of illustrations.

In Figure 6, an initial ellipse is drawn. Because other ellipses will share the same stroke-width (*3*), stroke color (*black*) and the same fill (*none*) , it is nestled inside a group (a `<g>` tag) so that it, and all other versions of it, may inherit these shared attribute values. So that the drawing will have vertical symmetry, the ellipse is then rotated 60 degrees, about its center (250,150), using the *transform* attribute:

```
<g  stroke="black" fill="none" stroke-width="3" >
   <ellipse id="E" ry="55" rx="100" cy="150" cx="250"
transform="rotate(60, 250,150)"/>
</g>
```

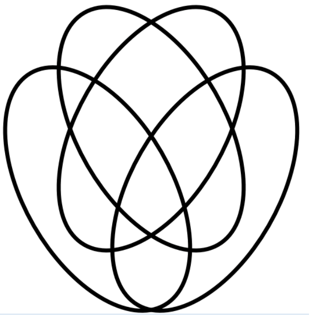


Fig. 6. The sixteen regions of a 4-Venn diagram drawn with congruent ellipses. The SVG version can be seen at `cs.sru.edu/~ddailey/svg/fourVenn1.svg`.

Next the ellipse is cloned using the `<use>` element, with the resulting code appearing as this:

```
<g  stroke="black" fill="none" stroke-width="3" >
  <ellipse id="E" ry="55" rx="100" cy="150" cx="250"
transform="rotate(60, 250,150)"/>
  <use xlink:href="#E" transform="translate(-40, 45)"/>
  <use xlink:href="#E" transform="rotate(60, 250,150)"/>
  <use xlink:href="#E" transform="translate(40, 45) ro-
tate(60, 250,150)"/>
</g>
```

The initial ellipse has been given an id (“E”) which can be referred to later (as #E in the *use* elements). There are three reuses of the ellipse named E: one is simply translated down 45 pixels and leftward 40 pixels. Another one is rotated an additional 60 degrees about its center. The final one is both rotated and translated. In SVG multiple transforms are performed from right to left, so that this last ellipse is first rotated 60 degrees from the original, and then repositioned into alignment. Another way of efficiently reusing code can be used in the construction of the radially symmetric 5-Venn diagram, shown in Figure 7, using the `<replicate>` tag, that is being considered for addition to SVG 2.0.

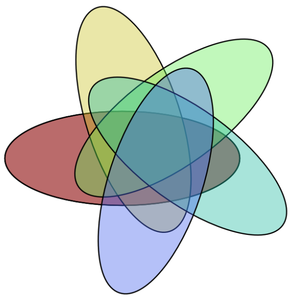


Fig. 7. A radially symmetric Venn diagram on 5 curves, drawn with only 10 lines of markup using <replicate>. The SVG version can be seen at `http://cs.sru.edu/~ddailey/svg/repVenn3.svg`

4.3 Paths

The use of SVG paths is particularly useful for drawing smooth curves with concavities. Using the `<path>` element, the diagram author has complete control over the slope of curves at the specified endpoints.

We will illustrate two uses of SVG paths, including the components for both quadratic and cubic Bézier curves, to draw a figure eight and a trefoil.

In a nutshell, SVG paths (see [11]) shape a curve using components based on lines, circular arcs, and Bézier components. The Bézier components allow us to draw, between any pair of points, a curve that visits both points and does so from any desired angle. Let us consider just a figure-eight curve such as shown in Figure 8, below (and at `http://cs.sru.edu/~ddailey/svg/figureEight.svg`).

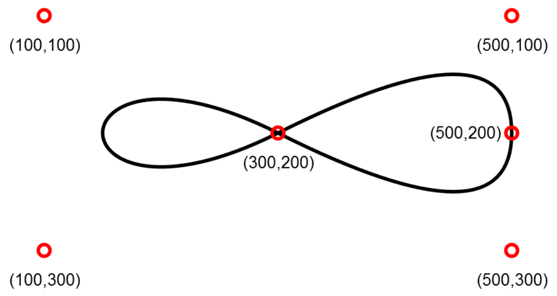


Fig. 8. Drawing a figure-eight using quadratic and cubic Bézier segments.

The code for this figure is as follows:

```
<path d="
  M 300,200
  Q 500,100 500,200
  Q 500,300 300,200
  C 100,100 100,300 300,200
"/>
```

Note that the curve begins at the point, 300,200, as marked and then progresses "in the direction" of the point 500, 100, finally ending up (after the first "Q") at 500,200. From there, it now heads south in the direction of 500,300, ultimately returning to 300, 200. That is, the two quadratic components of the curve are controlled by the collinear points (500,100) and (500, 300). The slope of the curve as it begins at (300,200) is ½ and, as it returns to (300,200) it is -½ . At the midpoint of its double quadratic swoop, the path is momentarily vertical. The cubic component while similar, uses two control points (chosen to be collinear with those of the quadratic component) so as to assure that its slopes at (300,200) are likewise ½ and – ½ . This demonstrates how we might stitch smooth segments together while controlling their slopes at key points such as intersections.

Circular arc components are conceptually a bit simpler, however, given two points and a radius of a circle that passes through both, there are, in fact, four circles that satisfy those constraints. Thus, the syntax for drawing arcs specifies with a few attributes which of those four curves is actually intended. (The interested reader is directed to the illustration at [12])

Without going into detail about this syntax, observe from Figure 9, that we may use circular arcs to adjoin line segments into smooth curves (in the sense of being continuously differentiable.)

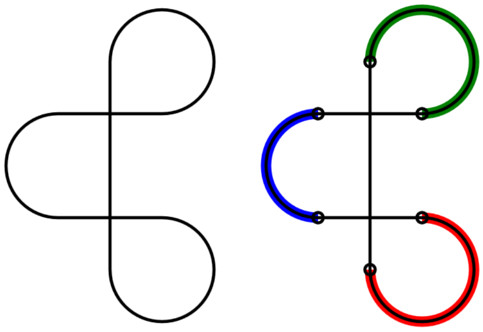


Fig. 9. A curve with three linear components and three circular components.

The curve at left in Figure 9 is drawn via the following code:

```
<path d="M 150,300   L 250 300
           A 50 50 0 1 1 200 350
          L 200 150
           A 50 50 0 1 1 250 200
          L 150 200
           A 50 50 0 0 0 150 300
"/>
```

while the curve at the right emphasizes the linear components (L) and the arc components (A) as well as the six connecting endpoints {(150,300), (250.300), (200,350), (200,150), (250,200), and (150,200)} .

Without much difficulty, one can interrupt this curve at key points of overlap to create a roadway with bridges as shown at `http://cs.sru.edu/~ddailey/svg/notknot3.svg`.

By redirecting some of the arcs of Figure 9, we may "reattach" some of those arcs, as shown in the following code to produce the shape on the left in Figure 10:

```
<path d="M 150,300   L 250 300
    A 100 100 0 0 0 340 245
    A 100 100 0 0 0 250 100
    A  50  50 0 0 0 200 150
  L 200 350
    A  50  50 0 0 0 250 400
    A 100 100 0 1 0 250 200
  L 150 200
    A 50 50 0 0 0 150 300
"   />
```

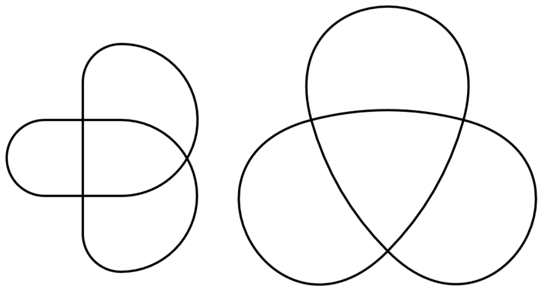


Fig. 10. Two versions of the trefoil. Left: using circular arcs; right: using Bézier curves.

The code for the diagram on the left of Figure 10 can then be generalized as shown in the diagram on the right, by using Bézier curves and making sure, as shown earlier in Figure 8, that the curve remains smooth.

```
<path d="M 200,200 Q 300, 174.12 400,200
C 600, 251.76 444.82,520.52 300,372.2
Q 227.59,299.54 200,200
C 145.82,0.82 455.18,0.82 400,200
Q 372.41,299.54 300,373.2
C 155.18,520.52 0,251.76 200,200
"/>
```

Traffic diagrams flowing along these curves, when drawn as knots, can be seen at `cs.sru.edu/~ddailey/svg/knot1.svg` and `cs.sru.edu/~ddailey/svg/knot2.svg` , while a similar illustration with Borromean rings can be seen at `cs.sru.edu/~ddailey/svg/rings.svg`.

5 Marking territory: shades, colors and patterns.

A cursory examination of Euler diagrams sampled in situ from the web, suggests that color (either as what SVG would call *fill* or *stroke*) is used to aid the viewer in "reading" the diagram. The concepts that such diagrams are used for (for example differentiating between the British Isles and the British Islands[1] [13], are frequently fairly complex (why else would we need a picture?), and as such helping the reader to parse the visual information seems to be one of the only uses of color (as articulated in Aesthetic Principles 9 and 10): namely to help recognize family traits (that subregions inherit from their regions) and to illustrate the individual differences (that distinguish subregions from their various containers, subsets and siblings).

However, color need not be the only way of doing this, and several centuries of printing with black ink have undoubtedly helped to inform the ways of "illustrating overlap" between geopolitical regions with and without the use of color. Techniques

[1] One might guess this to be a particularly difficult set of distinctions for American speakers of English.

of shading, coloring and patterning have been investigated by researchers in a broad array of fields (as instantiated by the visualization work of Ian McHarg and Edward Tufte). To what extent can some of these methods of conveying family resemblance and individual differences be expressed within SVG?

The simplest technique for combining markings of regions is to let their darknesses accumulate as more regions overlap, as shown in the code below and in Figure 11.

```
<g  stroke-width="5" stroke="black" fill-opacity=".4"
fill="black">
<circle cx="240" cy="60" r="50" />
<circle cx="290" cy="60" r="50"  />
<circle cx="265" cy="95" r="50"  />
</g>
```

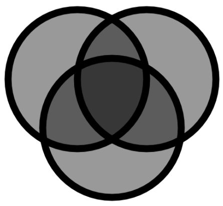
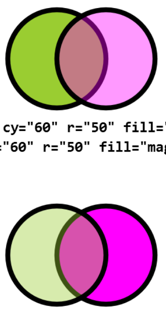


Fig. 11. Superimposition of partially opaque, simply shaded regions. The SVG version of this may be seen at `http://cs.sru.edu/~ddailey/svg/opacity3.svg`.

The problem with shading in this manner is that while it helps to differentiate "parent" regions from "child" regions, it does not help us distinguish things of the same generation. In SVG, objects maintain a stacking order from first to last declared. That is, two ovals, one magenta and one yellowgreen, and both 40% transparent, will not look the same, as in Figure 12. Thus, SVG imparts a bit of three dimensional asymmetry to its two dimensional objects.



```
1. <circle cx="560" cy="60" r="50" fill="yellowgreen"/>
<circle cx="610" cy="60" r="50" fill="magenta" fill-opacity=".4"/>
```

```
2.<circle cx="290" cy="270" r="50" fill="magenta"/>
<circle cx="240" cy="270" r="50" fill="yellowgreen" fill-opacity=".4"/>
```

Fig. 12. Effects of stacking order and opacity on appearance in SVG. SVG version may be seen at `http://cs.sru.edu/~ddailey/svg/opacity2.svg`.

Figure 13 investigates some combinations of color values and opacity values that seem to yield diagrams that are at least hint at satisfying principles 9 and 10. The obvious conclusion here is that the choice of colors and opacity values is somewhat critical toward producing a pleasant design, but it is not altogether obvious as to what might work. This is in part due to the fact that human color perception is complex. While most people have three differentially sensitive cones in the retina, some (mainly males) have only two, and some (mostly women) have recently been shown to have four. And while we are a predominantly trichromatic species at the level of the retina, we tend to be tetrachromats in the occipital lobe. Different cultures have differing numbers of primary color terms, though the boundaries between the primaries tend to coincide. Yellow is perceptually different than other primary or secondary colors and distance as calculated between two colors RGB values is not a good predictor of their perceptual distinctness. Couple all this with differing gamma curves for different monitors and the world of color is more complex than it might, at first glance, appear to be. Figure 13 also begins to raise and experiment, just a bit, with the interactions of labels and the regions underneath. Clearly, labels of regions must retain legibility, and that means that color contrasts must in some way be maximized, adding one more "readability" criterion to the Aesthetic Criteria already discussed.

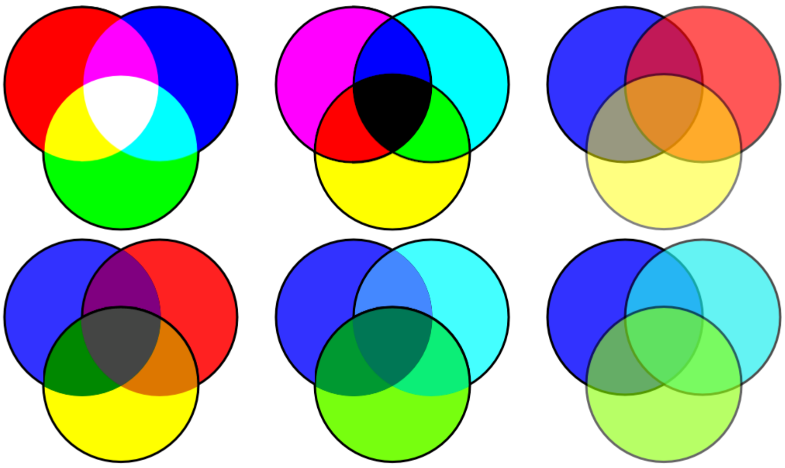


Fig. 13. Several differing color values offered to regions by varying the colors, opacities, and filtered combination of the original curves. (The original vector-based SSVG file can be found at `http://cs.sru.edu/~ddailey/svg/V9.svg`).

As mentioned earlier, SVG also has the ability to use filters to allow, among other things, the interaction of colors to be more precisely controlled, or to allow the intersections of regions to be precisely targeted. In Figure 14, one can observe that higher inter-region color differences can be promoted by either using multiplicative or screening filters (top left and top center) or by using compositions filters to choose specific regions and, to it, apply any chosen color whatsoever. That is, in browsers that support advanced filters (currently only the Adobe plugin for Internet Explorer, and Opera support these particular effects, though this is likely to change soon for both Firefox and Internet Explorer) we may exercise complete control over the coloration of the regions of a diagram, making the question of how to choose these colors all the more relevant.

While color is certainly a salient dimension of visual perception, it is not the only way to assign personality to shapes in SVG. Figure 14 shows several approaches using shapes and textures rather than just color to do so. These examples give rise quite naturally to a large set of complex questions about the distinguishability of various pattern glyphs, particularly in the context of multiple markings coalescing within regions that share features of two or more parent sets. That such information could be conveyed through tactile perception rather than through color, is a rather intriguing notion. While the meaningful primitives of color perception are relatively well studied, how might we most sensibly combine shapes in the manner? Is there a possibility of optimization is so complex a realm?

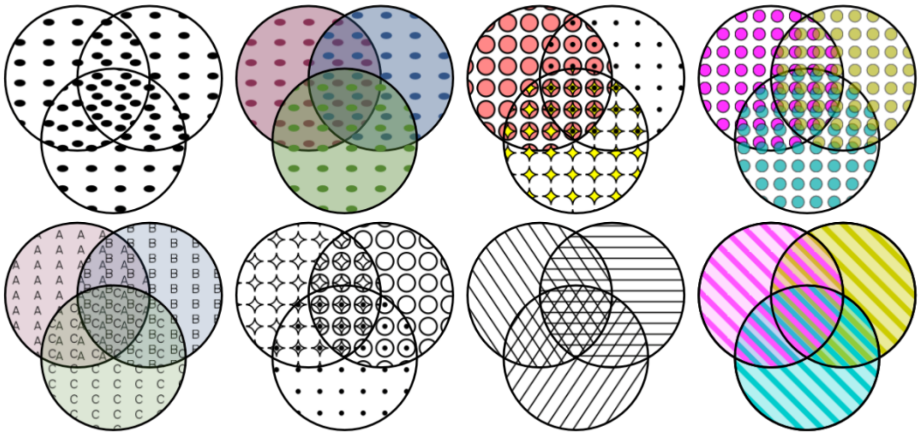


Fig. 14. Using SVG `<pattern>` to fill overlapping regions so as to preserve family traits. The SVG example can be studied at `http://cs.sru.edu/~ddailey/svg/V12.svg.` .

6 Conclusion

Interrelations between the technology of drawing and the aesthetics of diagrams have certainly occurred since the early days of printing. The progression from engraving to half-tones to photography, color printing and web-based vector graphics has been a fascinating one. An interdisciplinary topic such as how best to portray the fundamental inclusion relations among a set of concepts is a topic as old, no doubt, as philosophy. SVG offers promise to those engaged in these studies, across many disciplines since the language offers both succinctness of expression, broad and growing standards support, and access to high level tools such as CSS, JavaScript and server-side web tools. At the same time, maximizing control over our diagrams can, in some cases, require some of the most advanced features the language has to offer. Browser support for some of these features remains spotty, but shows continual improvement with new energy coming from major computing corporations as more parts of the SVG specification are implemented and as the specification itself shows very active growth and development.

References

1. Lilley, C.: W3C Scalable Vector Graphics (SVG) – History. W3C Interaction Domain, `http://www.w3.org/Graphics/SVG/History`
2. Dailey, D: An SVG Primer for Today's Browsers. W3C Working Draft — September 2010, `http://www.w3.org/Graphics/SVG/IG/resources/svgprimer.html` (2010)
3. Dailey, D., Elder, E., Perri, R.: A Browser-based Graphical User Interface for Designing and Manipulating Graphs. In: 7th Annual Conference, SVGOpen. W3C and SVG Working Group, Mountain View, CA `http://srufaculty.sru.edu/david.dailey/grapher/` (2009)
4. Bostock, M.: Voronoi Diagram,`http://mbostock.github.com/d3/ex/voronoi.html`
5. Dailey, D., Frost, J., Strazzullo, D.: Building Web Applications with SVG, Microsoft Press (2012)
6. Sowa, J.: Building, Sharing, and Merging Ontologies `http://www.jfsowa.com/ontology/ontoshar.htm`
7. Venn Diagram Survey: What is a Venn Diagram? The Electronic Journal Of Combinatorics (ed. June 2005), DS #5. `http://theory.cs.uvic.ca/~cos/venn/VennWhatEJC.html`
8. Micallef, L., Rodgers, P.: Drawing Area-Proportional Euler and Venn Diagrams using Ellipses, `http://www.eulerdiagrams.org/eulerAPE`
9. Dahlström, E., Dengler, P., et al.: Scalable Vector Graphics (SVG) 1.1 (Second Edition), W3C `http://www.w3.org/TR/SVG/` (2011)
10. Devaria, A.: When can I use...Compatibility tables for support of HTML5, CSS3, SVG and more in desktop and mobile browsers `http://caniuse.com/#cats=SVG`
11. Dailey D.: `<path>` in An SVG Primer for Today's Browsers, W3C (2010) `http://www.w3.org/Graphics/SVG/IG/resources/svgprimer.html#path`
12. Dailey, D.: `http://srufaculty.sru.edu/david.dailey/svg/newstuff/arcs.svg`
13. File:British Isles Euler diagram 15.svg, Wikipedia, `http://en.wikipedia.org/wiki/File:British_Isles_Euler_diagram_15.svg`

Introducing 3D Venn and Euler Diagrams

Peter Rodgers[1], Jean Flower[2], and Gem Stapleton[3]

[1] University of Kent, UK
p.j.rodgers@kent.ac.uk
[2] Autodesk, UK
[3] Visual Modelling Group, University of Brighton, UK
g.e.stapleton@brighton.ac.uk

Abstract. In 2D, Venn and Euler diagrams consist of labelled simple closed curves and have been widely studied. The advent of 3D display and interaction mechanisms means that extending these diagrams to 3D is now feasible. However, 3D versions of these diagrams have not yet been examined. Here, we begin the investigation into 3D Euler diagrams by defining them to comprise of labelled, orientable closed surfaces. As in 2D, these 3D Euler diagrams visually represent the set-theoretic notions of intersection, containment and disjointness. We extend the concept of wellformedness to the 3D case and compare it to wellformedness in the 2D case. In particular, we demonstrate that some data can be visualized with wellformed 3D diagrams that cannot be visualized with wellformed 2D diagrams. We also note that whilst there is only one topologically distinct embedding of wellformed Venn-3 in 2D, there are four such embeddings in 3D when the surfaces are topologically equivalent to spheres. Furthermore, we hypothesize that all data sets can be visualized with 3D Euler diagrams whereas this is not the case for 2D Euler diagrams, unless non-simple curves and/or duplicated labels are permitted. As this paper is the first to consider 3D Venn and Euler diagrams, we include a set of open problems and conjectures to stimulate further research.

1 Introduction

Euler diagrams represent intersection, containment and disjointness of sets. Currently, these diagrams are drawn in the plane and consist of labelled simple closed curves. These *2D Euler diagrams* have been widely studied over the last few years and much progress has been made on their theoretical underpinning and techniques for automatically drawing them.

Here we introduce the concept of *3D Euler diagrams*. We know of no other work defining this type of 3D representation and, thus, this paper focusses on setting the groundwork for discussing this new diagrammatic type. Furthermore, it provides a platform to engage the community in discussion about the various issues in 3D Euler diagram research. 3D Euler diagrams consist of labelled orientable closed surfaces drawn in $\mathbb{R}^3$. An example of a 2D and a 3D Euler diagram representing the same information can be seen in figure 1. This 3D diagram, as well as all of the 3D Euler diagrams drawn in this paper, can be accessed from

3rd International Workshop on Euler Diagrams, July 2, 2012, Canterbury, UK.

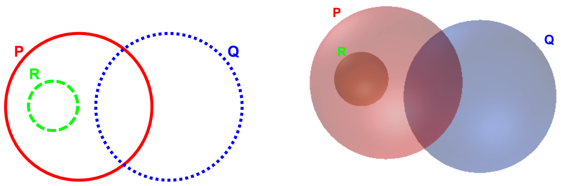


Fig. 1. A 2D Euler diagram with an equivalent 3D Euler diagram.

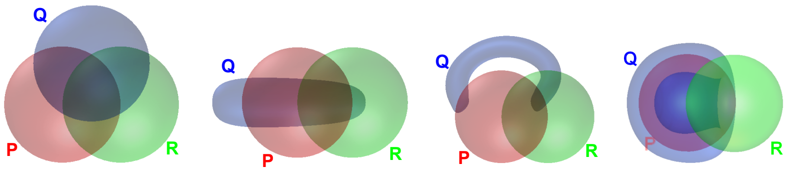


Fig. 2. Four topologically distinct wellformed Venn-3s.

`www.eulerdiagrams.com/3D/workshop/`. Using the freely available Autodesk Design Review software, one can rotate and explore the 3D diagrams.

We define *3D Venn diagrams* as 3D Euler diagrams where all combinations of surface intersections are present. An interesting comparison between 2D and 3D is in the common Venn-3 case, i.e the Venn diagram representing exactly three sets. It is known that there is only one topologically distinct embedding of wellformed Venn-3 in 2D [9]. In 3D, there are infinitely many topologically distinct embeddings of wellformed Venn-3 when the surfaces are closed and orientable (i.e. connected sums of tori). When the surfaces are topologically equivalent to the sphere, there are at least four topologically distinct embeddings of wellformed 3D Venn-3, shown in figure 2.

Whilst 3D Venn and Euler diagrams are interesting in their own right, we believe that there are also solid practical motivations for examining them. Firstly, the recent advances in hardware available for 3D display and interaction (eg. 3D televisions and Microsoft Kinect) support 3D visualization. As Venn and Euler diagrams form an important aspect of 2D visualization, it is reasonable to expect that they will also be important for 3D visualization.

Secondly, there are intrinsic benefits to exploring 3D with respect to Euler diagrams. When 2D Euler diagrams are defined as consisting of (anything equivalent to) simple closed curves without duplicated labels (for instance [4] and [6]), not all data sets can be visualized. This is clearly a major limitation. Subsequently, the definition of a 2D Euler diagram was relaxed, permitting diagrams to have non-simple curves and duplicated curve labels [7, 10]. Under this new approach, all data sets can be visualized but potentially at the cost of significantly reduced usability [8]. However, as we note later in this paper, in the 3D case we conjecture that it is possible to draw all data sets (encapsulated by *diagram descriptions*) without duplicate labels and non-simple surfaces. Consequently, this major limitation on undrawability is overcome in the 3D case.

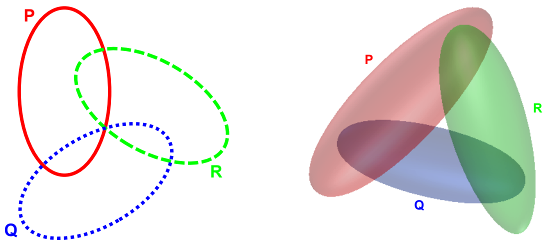


Fig. 3. An non-wellformed 2D diagram and an equivalent wellformed 3D diagram.

Wellformedness properties are a key aspect of drawing of Euler diagrams. In 2D, they relate to how the curves intersect and to the properties of the regions present. In 3D, we generalize them to how the surfaces intersect and the properties of the solids to which the surfaces give rise. The 2D Euler diagram on the left of figure 3 is not wellformed because it has a triple point of intersection between the curves. By contrast, the same data can be represented in a wellformed manner in 3D, as shown in the righthand side of figure 3; we will demonstrate, in section 4, that any way of drawing a 2D Euler diagram representing the same data breaks a wellformedness property. Often, there exists wellformed 3D Euler diagrams for data that has no wellformed 2D representation.

The remainder of this paper is as follows. Section 2 formally defines 3D Euler diagrams and related concepts. Section 3 generalizes wellformedness properties of 2D Euler diagrams to the 3D context. Section 4 establishes that more data sets can be drawn wellformed with 3D Euler diagrams than with 2D Euler diagrams. We then go on to examine future work and propose open questions in section 5. Finally, section 6 concludes.

2 What is a 3D Euler Diagram?

3D Euler diagrams are formed from closed surfaces embedded in $\mathbb{R}^3$ rather than closed curves embedded in $\mathbb{R}^2$. We refer the reader to [12] for a formal definition of a 2D Euler diagram and associated wellformedness properties. As with closed curves in 2D Euler diagrams, which are typically required to be simple, we choose not to use arbitrary surfaces in 3D Euler diagrams. This is because we want to be able to define certain properties of 3D Euler diagrams that require the surfaces to be 'nice'. Choosing our surfaces to be orientable gives us a well-understood notion of what constitutes the interior. Hence, we define 3D Euler diagrams as follows, where $\mathcal{L}$ is a set of labels that we use to label the surfaces:

Definition 1. *A* ***3D Euler diagram*** *is a pair, $d = (\mathcal{S}, l)$, where*

1. *$\mathcal{S}$ is a finite set of closed, orientable surfaces embedded in $\mathbb{R}^3$, and*
2. *$l : \mathcal{S} \to \mathcal{L}$ is an injective function that labels each surface.*

In 2D Euler diagrams, zones are sets of points in the plane that are inside all curves in a given set and outside the rest of the curves in the diagram. In figure 1, both diagrams have five zones. Zones are fundamentally important, since these correspond to the semantics of the diagram: between them, the present zones must represent all of the non-empty set intersections. We now generalize the notion of a zone to the 3D case:

Definition 2. *A **zone** in a 3D Euler diagram, $d = (\mathcal{S}, l)$, is a set of points, z, in $\mathbb{R}^3$ for which there exists a subset, S, of $\mathcal{S}$ such that*

1. *every point, p_{in}, in z is inside all of the surfaces in S and outside all of the surfaces in $\mathcal{S} - S$, and*
2. *z is maximal with this property.*

Such a zone, z, is described by $des(z) = \{l(s) : s \in S\}$. The set of zones in d is denoted $\mathcal{Z}(d)$.

In the visualization process, one starts with a description of the to-be-drawn diagram. A diagram description is a list of the set intersections that must be present in the diagram, given the sets to be visualized, thus precisely encapsulating the categories in which data items lie. For example, suppose we wish to visualize the sets P, Q, and R, and the intersections we wish to visualize are $P \cap \overline{Q} \cap \overline{R}$, $Q \cap \overline{P} \cap \overline{R}$, $R \cap \overline{P} \cap \overline{Q}$ (i.e. the set intersections that comprise elements in exactly one of the three sets), along with $P \cap Q \cap \overline{R}$, $P \cap R \cap \overline{Q}$ and $Q \cap R \cap \overline{P}$ (i.e. the set intersections that comprise elements that are in exactly two of the sets). Further, we also must visualize the set intersection that comprises elements in none of the three sets, namely $\overline{P} \cap \overline{Q} \cap \overline{R}$. This is more succinctly represented as $\emptyset, P, Q, R, PQ, PR, QR$, listing the non-complemented sets from each specified intersection, and is visualized by both diagrams in figure 3. More formally these diagrams have description $\{\emptyset, \{P\}, \{Q\}, \{R\}, \{P, Q\}, \{P, R\}, \{Q, R\}\}$, but we will abuse notation as just illustrated.

Definition 3. *A **diagram description**, D, is a subset of $\mathbb{P}\mathcal{L}$ that includes $\emptyset$. The **description** of a 3D Euler diagram, $d = (\mathcal{S}, l)$, is $\{des(z) : z \in \mathcal{Z}(d)\}$.*

The classic drawing problem, generalized to 3D, is *given a diagram description, D, draw a 3D Euler diagram with description D*. In 2D, this problem is often subject to a range of extra constraints that typically relate to the wellformedness properties. For instance, we may wish to find a diagram that has no concurrency between surfaces. We generalize the wellformedness properties to 3D in the next section.

3 Wellformedness Properties of 3D Euler Diagrams

There are various wellformedness properties that can be applied to 2D Euler diagrams [12]. These are informally described in table 1, where we also present their generalizations to 3D. Examples of non-wellformed diagrams in both 2D

Property	2D Case	3D Case
Connected Zones	Every zone is a connected component of $\mathbb{R}^2$.	Every zone is a connected component of $\mathbb{R}^3$.
n-Point	Every point in $\mathbb{R}^2$ is passed through at most $n = 2$ times by the curves.	Every point in $\mathbb{R}^3$ is passed through at most $n = 3$ times by the surfaces.
Crossings	Whenever two curves intersect, they cross transversely.	Whenever two surfaces intersect, they cross transversely.
Line Concurrency	No two curves share a common line segment.	No three surfaces share a common line segment.
Surface Concurrency	N/A	No two surfaces share a common sub-surface.

Table 1. Wellformedness properties.

and 3D are shown in table 2. Some of the wellformedness properties in 3D are obvious generalizations of the 2D case, but others benefit from further discussion.

First, consider the n-points properties. For the 2D case, a diagram is non-wellformed if it contains a triple point (i.e. a 3-point). The reason that the presence of 2-points does not render a diagram non-wellformed is because whenever two curves intersect, a 2-point is formed. However, given three curves that pairwise intersect it need not be the case that a 3-point is formed. Thus, 3-points are avoidable in 2D. However, 3-points are not avoidable in 3D. This is illustrated in figure 4. Here, three spheres intersect to form Venn-3. The cross-section of the diagram shown on the right of the figure illustrates that a three 3-point is formed.

Now consider now line concurrency. In 2D, diagrams that have two curves running concurrently along a line segment are not wellformed. However, in 3D, such a property is unavoidable: two surfaces that intersect and share common interior points necessarily share a common line segment. For instance, the Venn-3 diagram drawn with spheres on the left of figure 4 has line concurrency.

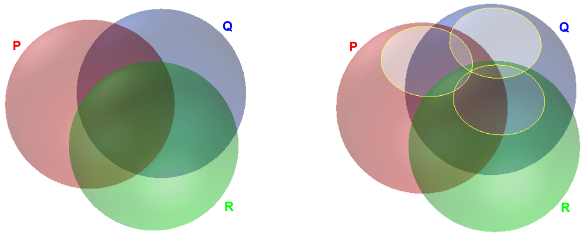


Fig. 4. The necessity of 3-points in 3D.

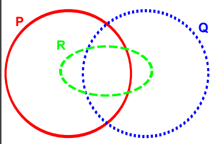
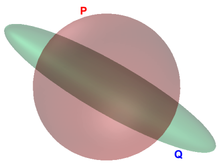
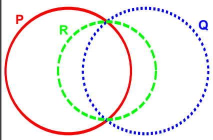
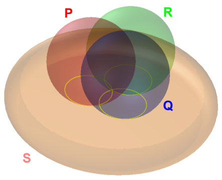
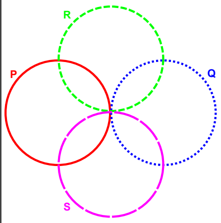
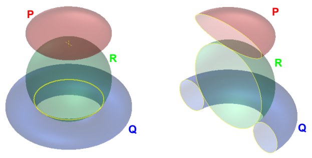
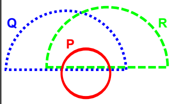
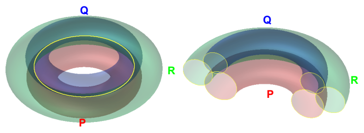
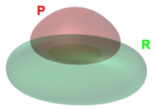
Property	2D Case	3D Case
Connected Zones	 The zone PQ is disconnected.	 Here, P is a sphere with a 'sausage', Q, through it. The zone inside Q and outside P is disconnected.
n-point	 The curves P, Q, and R form two 3-points.	 The spheres P, Q, and R form a 4-point where they all intersect with S.
Crossings	 The curves P and Q intersect at a point where they do not cross (as do R and S).	 The sphere R intersects with Q but does not cross Q; a cross-section is shown on the right.
Line Concurrency	 The two curves P and Q share a common line segment.	 The three tori share a common line segment; a cross-section is shown on the right.
Surface Concurrency	N/A	 The two 'squashed' spheres share a disc-like surface.

Table 2. Examples of non-wellformed diagrams.

4 Drawability of 3D Euler Diagrams

One of our key motivations for developing 3D Euler diagrams is that they allow more diagram descriptions to be drawn in a wellformed manner than is the case for 2D Euler diagrams. Recall that a 2D Euler diagram comprises a set of labelled simple closed curves such that no label is used on more than one curve.

Definition 4. *Let D be a diagram description. Then D can be* ***drawn wellformed*** *if there exists a Euler diagram with description D which satisfies all of the wellformedness properties of table 1.*

We now establish that every diagram description that can be drawn wellformed in 2D can be drawn wellformed in 3D. Our proof strategy is to convert a wellformed 2D diagram into a wellformed 3D diagram with the same description. To illustrate the approach, consider figure 5. Here, the wellformed 2D diagram is converted into a 3D diagram by rotating the 2D diagram around line that does not pass through any of the curves. Each resulting surface is a torus and the final 3D diagram is shown on the right.

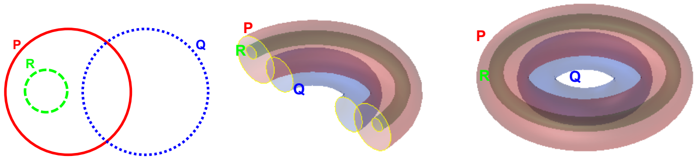


Fig. 5. Converting a wellformed 2D diagram into a wellformed 3D diagram.

Theorem 1. *Let D be a diagram description. If D can be drawn wellformed in 2D then D can be drawn wellformed in 3D.*

Proof (Sketch). Suppose that D can be drawn wellformed in 2D. Choose any wellformed 2D diagram, d_2, with description D. Draw a line, λ, that does not pass through any curve in d_2. Rotate d_2 about λ by 2π to create a 3D Euler diagram, d_3. Each closed curve, c_2, in d_2 gives rise to a torus, t_3, in d_3 and we label t_3 the same as c_2. It can be shown that each zone, z_2, in d_2 gives rise to a zone, z_3, in d_3 with the same description and that no other zones appear in d_3. That is, the description of d_3 is D. The wellformedness of d_3 can be trivially established using the wellformedness of d_2.

We now demonstrate that there are diagram descriptions that cannot be drawn wellformed in 2D that can be drawn wellformed in 3D. Work by Flower and Howse [4] identified necessary and sufficient conditions for when a diagram description can be drawn wellformed in the 2D case. We will demonstrate that their conditions are not both necessary and sufficient in 3D, failing in multiple ways. Their approach starts by converting a diagram description into a graph, called the *super-dual*, and looks at properties of this graph to establish drawability.

Definition 5. *Given a diagram description, D, the* ***super-dual*** *of D is a graph, $G = (V, E)$, where $V = D$ is the set of vertices and, for every pair of vertices, v_1 and v_2, there is an edge between v_1 and v_2 if and only if v_1 and v_2 differ by a single label (recall elements of D, i.e. the vertices, are sets of labels).*

Assuming we have a super-dual that is planar, we can draw that graph in the plane without edges crossing. Given such an embedding of the super-dual, we can attempt to form the required Euler diagram. An illustration of the process is given in figure 6. Here, we start with description $\emptyset, P, Q, PQ$ and turn it into the super-dual, shown on the left of figure 6. The curves of the 2D Euler diagram are constructed by enclosing the vertices appropriately. For example, to draw a curve labelled P we enclose the vertices that include P but no others. The curve labelled Q is similarly formed. Finally, we delete the super-dual and are left with the required 2D Euler diagram, shown on the right.

The preceding example is rather simple, but it is by examining the super-dual and, if necessary, its subgraphs that we can determine wellformed drawability in the 2D case. Now, the curves of a (wellformed) 2D Euler diagram are all simple which means that for each curve, c, the set of points inside c is a simply connected region. In terms of a super-dual, this implies that the maximal subgraph induced by the vertices that contain the label of c is connected and, moreover, that the subgraph induced by the vertices that do not contain the label of c is also connected. This key insight led Flower and Howse to define the *connectivity conditions* for graphs; these are used to establish properties of super-duals arising from diagram descriptions.

Definition 6 (Connectivity Conditions [4]). *Let $G = (V, E)$ be a graph such that $V \subseteq \mathbb{P}\mathcal{L}$. The* ***connectivity conditions*** *for G are:*

1. *G is connected,*
2. *for each curve label, λ, in $\mathcal{L}$, the maximal subgraph of G whose vertices include λ is connected, and*
3. *for each curve label, λ, in $\mathcal{L}$, the maximal subgraph of G whose vertices do not include λ is connected.*

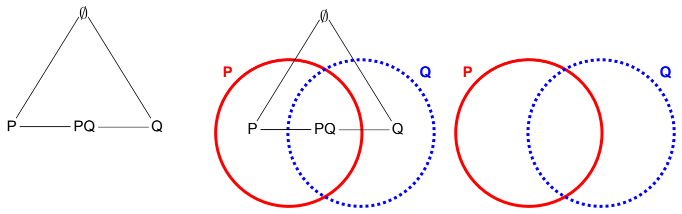


Fig. 6. Constructing a 2D Euler diagram from the super-dual.

Theorem 2 (2D Connectivity Test [4]). *Let D be a diagram description whose super-dual fails connectivity conditions. Then there is no 2D Euler diagram, d, with description D that is wellformed.*

The connectivity conditions are necessary for wellformed drawability in 3D:

Theorem 3 (3D Connectivity Test). *Let D be a diagram description whose super-dual fails connectivity conditions. Then there is no 3D Euler diagram, d, with description D that is wellformed.*

Flower and Howse further introduce the *face conditions*, which we now informally explain via an example; for full details we refer to [4]. Consider the diagram description $\emptyset, P, Q, PQ, R, PR, QR$. This has super-dual as shown on the left of figure 7. All three curves will pass through the face f, which will lead either to a 3-point (as shown in the figure), a disconnected zone, or an un-required zone (to create such a zone, nudge one of the curves to remove the triple point). The non-wellformedness of the diagram is determined by examining the edges around f. By traversing the simple cycle around f, each time we pass along an edge we write down the curve label that is in one of the incident vertices but not the other to form a word, say $w = RPQRPQ$. By examining alternations of letters in w, we can see which curves are required to cross. For instance, P and Q alternate, since $PQPQ$ is a scattered subword of w. This tells us that (the curves labelled) P and Q must cross in f. Similarly, P and R must cross and Q and R must cross. This indicates the possible presence of a triple point. In general, a combinatorial analysis of the words around faces in the graph is used to determine whether the plane embedding will give rise to a wellformed 2D Euler diagram. The face conditions for the graph, roughly speaking, identify whether too many crossings occur for wellformedness to be achieved. Of note is that our example is very simple and the actual details are more complex than we have illustrated. In any case, the super-dual in figure 7 is planar, passes the connectivity conditions, but fails the face conditions. Hence, this embedding of the super-dual cannot be used to draw a wellformed 2D Euler diagram with the specified description. Moreover, there is no different choice of embedding which passes the face conditions.

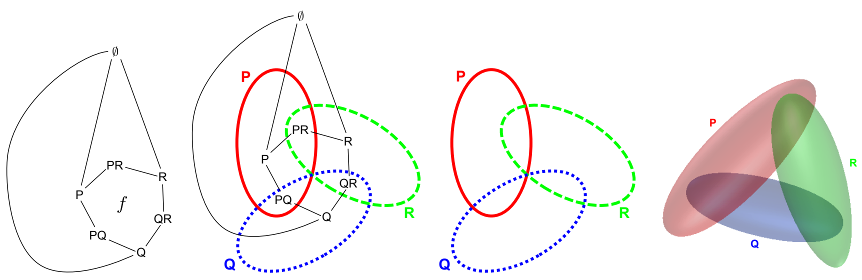


Fig. 7. Failure of the face conditions.

For some diagram descriptions, but not the one just considered, it is possible to remove edges from the super-dual whilst ensuring connectivity holds and produce a subgraph, G, that has an embedding which passes the face conditions. A further complication is the potential lack of planarity of the super-dual. Again, we may be able to remove edges to create a planar subgraph G with the properties just described. In either case, if such a G exists then D is drawable as a wellformed 2D Euler diagram, otherwise it is not. This key result is captured in the following theorem:

Theorem 4 (2D Drawability – Necessary and Sufficient Conditions [4]). *Let D be a diagram description with super-dual $SG(D)$. There exists a wellformed 2D Euler diagram that is a drawing of D iff there exists a planar subgraph, G, of $SG(D)$ obtained by removing edges from $SG(D)$, which passes the connectivity conditions and has a plane embedding that passes the face conditions.*

We can immediately generalize one side of this theorem to the 3D case:

Theorem 5 (3D Drawability – Sufficient Conditions). *Let D be a diagram description with super-dual $SG(D)$. If there exists a planar subgraph, G, of $SG(D)$ obtained by removing edges from $SG(D)$, which passes the connectivity conditions and has a plane embedding that passes the face conditions then there exists a wellformed 3D Euler diagram that is a drawing of D.*

Proof. By theorem 4, a wellformed 2D Euler diagram exists. The result then follows from theorem 1.

We now demonstrate that there are diagram descriptions that are not drawable wellformed in 2D (they fail one of more of the conditions in theorem 4) but that are drawable wellformed in 3D. The three examples below fail the conditions of Theorem 4 in different ways. This shows that there are more diagram descriptions that can be drawn wellformed in 3D than 2D and that the conditions to determine drawability in 2D are not useful for determining drawability in the context of 3D diagrams. For these three examples, the wellformedness of the 3D representations suggests that they are more readable than the 2D representations and the 3D diagrams display a pleasing symmetry.
Example 1: Figure 7 shows a planar super-dual that passes the connectivity conditions but fails the face conditions, so the corresponding 2D representation shown in figure 7 is not well-formed (it has a triple-point). All plane embeddings

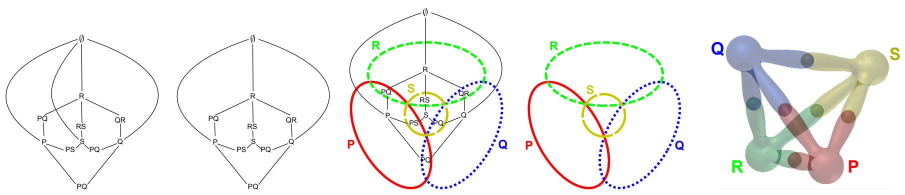


Fig. 8. Failure of planarity of the super-dual.

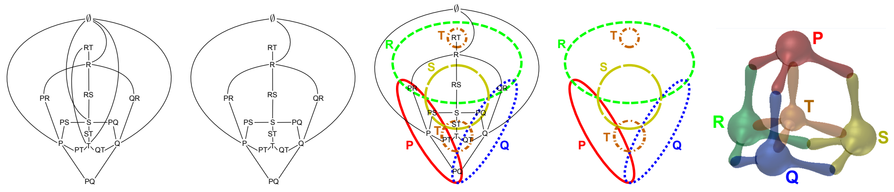


Fig. 9. Failure of planarity with no planar subgraph that passes connectivity.

of this graph fail the face conditions. Removing edges from this graph will never result in a graph that passes the connectivity and face conditions. Hence, there is no wellformed 2D Euler diagram with the given description. A wellformed 3D Euler diagram with this description can be seen in figure 7.

Example 2: Figure 8 shows a super-dual that is non-planar, since it is homeomorphic to $K_{5,5}$, so some edge removal is necessary to achieve planarity. We demonstrate that any way in which edges can be removed to achieve planarity whilst maintaining connectivity does not produce a graph which passes the face conditions. If we remove an edge from the super-dual that is not incident to $\emptyset$ then we break the connectivity conditions. If we remove an edge which is incident to $\emptyset$ then the graph becomes planar and connectivity is preserved, so we then look for an embedding which passes the face conditions. However, any plane embedding of the graphs resulting from the removal of exactly one of these edges, shown here with the edge between $\emptyset$ and S removed, fails the face conditions. Continuing this kind of analysis, it can be demonstrated that there is no wellformed 2D Euler diagram with the given description. A wellformed 3D Euler diagram with the same description can be seen in figure 8.

Example 3: Figure 9 shows a super-dual that is non-planar, since it has a proper subgraph homeomorphic to $K_{5,5}$, so again some edge removal is necessary to achieve planarity. However, all planar subgraphs fail the connectivity conditions. Take one edge as an example, say the edge between T and RT. This edge is in the maximal subgraph of the super-dual whose vertices include T. The removal of the T-RT edge would disconnect this subgraph, breaking connectivity. The same argument prevents removal of any edge not incident to $\emptyset$. Thus, the only edges we can consider removing are those incident with $\emptyset$. However, the subgraph obtained by removing the vertex $\emptyset$ is homeomorphic to $K_{5,5}$. This implies that we cannot obtain a planar subgraph by removing edges from the super-dual whilst preserving connectivity. Hence, there is no wellformed 2D Euler diagram with the given description. A wellformed 3D Euler diagram with the same description can be seen in figure 9.

Thus, more diagram descriptions are drawable wellformed in 3D than in 2D. In particular, the face conditions need not be passed in order for us to have wellformed drawability in 3D and we need not have planarity of the dual.

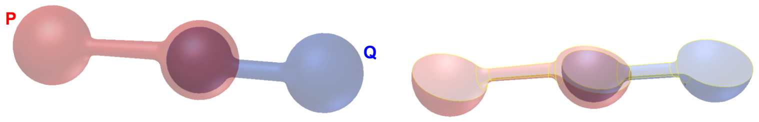


Fig. 10. Constructing 3D diagrams from descriptions.

5 Future Work and Open Problems

There are numerous open questions in 3D. Following the previous section:

Open Problem 1 *What are necessary and sufficient conditions for determining wellformed drawability in the 3D case?*

We have demonstrated that the connectivity conditions are necessary, but there is no obvious generalization of the face conditions to the 3D case (the notion of a face does not translate to 3D graphs). We conjecture that a different approach is needed and it is very possible that this could provide a new perspective on wellformed drawability in the 2D case as well.

An important question is how the definition of Euler diagrams needs to be relaxed in order to draw every diagram description. As discussed previously, in 2D we require either non-simple curves or duplicated label use. We believe that the definition given in this paper for the 3D case is sufficient for drawability in general, if we do not impose any wellformedness properties:

Conjecture 1 *For every diagram description there exists a 3D Euler diagram with that description.*

We are confident that this conjecture is true because we believe the following method for construction works in general. Given the diagram description $D = \{\emptyset, \{P\}, \{Q\}, \{P, Q\}$ for each element, z, in D create one sphere for each label in z and draw them concurrently. For each pair of elements, z_1 and z_2, in D, if they share a non-empty set of labels, $L = z_1 \cap z_2$, then join the spheres with labels in L arising from z_1 and z_2. This example can be seen in figure 10.

We focus now on a specific class of Euler diagrams which is the widely known family of Venn diagrams [9]. In 2D, Venn diagrams are Euler diagrams where all 2^n possible intersections between n sets are represented by connected regions, that is there are 2^n zones each of which is connected. In 3D:

Definition 7. *A* ***3D Venn diagram****,* $d = (\mathcal{S}, l)$*, is a 3D Euler diagram where there are* $2^{|\mathcal{S}|}$ *zones, each of which is connected.*

Four topologically distinct embeddings of Venn-3 are shown in the introduction, figure 2. To see that they are distinct, we make arguments about their zones. The first (leftmost) Venn-3 has only simply connected zones. The second Venn-3 has exactly two zones that not simply connected, namely P and PR. The third Venn-3 has exactly two zones that are not simply connected, namely

$\emptyset$ and R. The fourth (rightmost) Venn-3 also has exactly two zones that are not simply connected, namely QR and PQR. The four diagrams are pairwise topologically distinct because the non-simply connected zones are contained by different numbers of surfaces.

Conjecture 2 *There are exactly four topologically distinct embeddings of well-formed 3D Venn-3 when the surfaces are all topologically equivalent to spheres.*

A variety of other open problems can be stated for 3D Venn diagrams, some of which have been answered for 2D Venn diagrams (see [9] for an excellent survey on results for 2D Venn diagrams). One such example is:

Open Problem 2 *How many topologically distinct embeddings of wellformed Venn-n exist when the surfaces are all topologically equivalent to spheres?*

Returning to the more general case of Euler diagrams, there has been considerable interest in drawing them with curves of particular shapes. For instance, Stapleton et al. [13] identified a class of diagram descriptions that could be drawn using only circles and Wilkinson devised a method that only drew Euler diagrams using circles [14]. Kestler et al. devised a method for drawing Euler diagrams with regular polygons [5] and others have considered drawing Venn diagrams where the curves have other geometric shapes, such as triangles [1]. Thus, curve shape is considered interesting and important in the 2D case. For 3D, this generalizes to surface shape (where we no longer mean 'up to topological equivalence'). We pose the following two problems concerning surface shape:

Open Problem 3 *What class of diagram descriptions can be drawn when the surfaces are all some specified shape, such as spheres?*

Open Problem 4 *Can all diagram descriptions that can be drawn wellformed in 2D using only circles can be drawn wellformed in 3D using only spheres?*

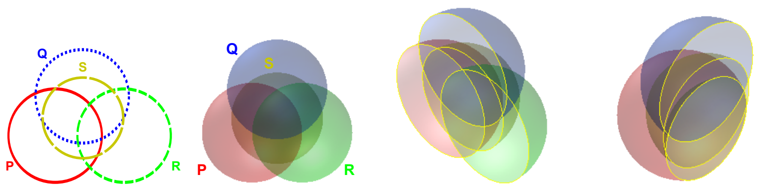


Fig. 11. Converting from circles to spheres.

A naïve method for converting a diagram drawn with circles into one drawn with spheres is to use each circle to generate a sphere. However, this construction approach need not lead to the required diagram description or preserve wellformedness. This is demonstrated in figure 11: the leftmost Euler diagram is drawn with circles, the 3D Euler diagram is obtained by converting the circles to spheres and the remaining diagrams show cross-sections of the 3D Euler diagram. Unfortunately, this created an extra zone, that inside only S and, moreover, this zone is disconnected. We conjecture that there does not exist a wellformed 3D diagram drawn with spheres with the same diagram description as figure 11. However, we believe that some classes of diagram descriptions drawable wellformed with circles can be drawn wellformed with spheres:

Conjecture 3 *The class of inductively pierced descriptions, introduced in [13] and generalized in [10], which can all be drawn wellformed with circles in 2D can be drawn wellformed with spheres in 3D.*

Finally, there has also been significant interest in drawing 2D Euler diagrams in a so-called *area-proportional* manner. In the area-proportional 2D case, the zones must have specified areas. Key publications on area-proportional Venn and Euler diagram drawing include Chow and Rodgers [2], Chow and Ruskey [3], Kestler et al. [5], Stapleton et al. [11], and Wilkinson [14]. These methods mostly consider drawing the diagrams where the curves have specific shapes, such as circles. In 3D, the area-proportional case generalizes to the zones having specified volumes, the *volume-proportional* case.

Definition 8. *A* ***volume specification*** *is a function, $v\colon \mathbb{P}\mathcal{L} - \{\emptyset\} \to \mathbb{R}^+ \cup \{0\}$. The diagram description* ***induced*** *by v is $\{z : v(z) \neq \emptyset \vee z = \emptyset\}$. A 3D Euler diagram* ***conforms*** *to a volume specification, v, if its description is induced by v and its zones have the volumes specified by v.*

Open Problem 5 *What class of volume specifications can be drawn in a wellformed manner?*

Open Problem 6 *What class of volume specifications can be drawn where the surfaces are all some specified shape?*

6 Conclusion

In this paper we have introduced the concept of 3D Euler diagrams, formally defining them as orientable closed surfaces which implies the surfaces are simple. We have compared them with 2D Euler diagrams and discovered that 3D Euler diagrams have some benefits over 2D Euler diagrams in terms of drawability when wellformedness is considered. In particular, we have shown that there are more diagram descriptions that can be drawn wellformed with 3D diagrams than 2D diagrams and we conjecture that all diagram descriptions can be drawn in 3D without allowing non-simple surfaces and duplicate label use, unlike in 2D.

We established that there are four topologically distinct wellformed embeddings of 3D Venn-3, whereas there is only one such embedding in 2D. This demonstrates that there is more choice in terms of how we layout diagrams in 3D over 2D, which is likely to be beneficial when more sets need to be represented. This gives further insight into why more diagram descriptions can be drawn wellformed in 3D: we have greater control over which zones are topologically adjacent and topological adjacency impacts whether we can add a new surface (or curve in 2D) and maintain wellformedness.

By presenting a series of open questions and conjectures, we hope to stimulate research progress on 3D Euler diagrams. In some cases there is no obvious way to extend existing 2D results to the 3D case, such as Open Problem 1 concerning the drawability of wellformed diagrams. Hence, a different approach is likely to be required. It may be that results in the 3D case will allow more progress to be made in the 2D case.

With the advent of recent affordable 3D display, interaction and printing devices, 3D visualization has the potential to be commonplace. We expect that 3D Euler diagrams will form a useful component in this field.

Acknowledgement Gem Stapleton was partially supported by an Autodesk Education Grant.

References

1. J. Carroll. Drawing Venn triangles. Technical report, HP Labs HPL-2000-73, 2000.
2. S. Chow, P. Rodgers. Constructing area-proportional Venn and Euler diagrams with three circles. In *Euler Diagrams 2005*.
3. S. Chow, F. Ruskey. Towards a general solution to drawing area-proportional Euler diagrams. In *Euler Diagrams 2004*, *ENTCS*, pages 3–18, 2005.
4. J. Flower, J. Howse. Generating Euler diagrams. In *Diagrams 2002*, pp 61–75. Springer, 2002.
5. H. Kestler, A. Muller, T. Gress, M. Buchholz. Generalized Venn diagrams: A new method for visualizing complex genetic set relations. *Journal of Bioinformatics*, 21(8):1592–1595, 2005.
6. O. Lemon, I. Pratt. Spatial logic and the complexity of diagrammatic reasoning. *Machine GRAPHICS and VISION*, 6(1):89–108, 1997.
7. P. Rodgers, L. Zhang, A. Fish. General Euler diagram generation. In *Diagrams 2008*, pp 13–27. Springer, 2008.
8. P. Rodgers, L. Zhang, H. Purchase. Wellformedness properties in Euler diagrams: Which should be used? *accepted for IEEE Transactions on Visualization and Computer Graphics*, 2012.
9. F. Ruskey. A survey of Venn diagrams. *Electronic Journal of Combinatorics*, 1997. www.combinatorics.org/Surveys/ds5/VennEJC.html.
10. G. Stapleton, J. Flower, P. Rodgers, J. Howse. Automatically drawing Euler diagrams with circles. *Journal of Visual Languages and Computing*, accepted 2012.
11. G. Stapleton, P. Rodgers, J. Howse A general method for drawing area-proportional Euler diagrams. *Journal of Visual Languages and Computing*, 22(6):426–442, 2011.
12. G. Stapleton, P. Rodgers, J. Howse, J. Taylor. Properties of Euler diagrams. In *Layout of Software Engineering Diagrams*, pp 2–16. EASST, 2007.
13. G. Stapleton, L. Zhang, J. Howse, P. Rodgers. Drawing Euler diagrams with circles: The theory of piercings. *IEEE Transactions on Visualisation and Computer Graphics*, 17(7):1020-1032, 2011.
14. L. Wilkinson. Exact and approximate area-proportional circular Venn and Euler diagrams. *IEEE Transactions on Visualization and Computer Graphics*, available online, 2011.

FunEuler: an Euler Diagram based Interface Enhanced with Region-based Functionalities

Gennaro Cordasco[1], Rosario De Chiara[2], Andrew Fish[3⋆], and Vittorio Scarano[2]

[1] Dipartimento di Psicologia - Seconda Università di Napoli - ITALY
[2] ISISLab - Dipartimento di Informatica - Università di Salerno - ITALY
[3] School of Computing, Mathematical and Information Sciences, University of Brighton - UK

Abstract. Euler Diagrams are an accessible means of representing non hierarchical set-based relationships which have recently been used in resource management interfaces to facilitate user categorisation. We develop a novel, extensible Euler diagram based interface, called FunEuler, which integrates the concepts of visual classification, spatial arrangements and functional application, thereby greatly extending the power of such Euler diagram based interfaces by enabling fast application of a collection of predefined functions to collections of categorised resources. To demonstrate the principle, we provide several functionalities such as file zipping or creating playlists within the application, whilst also providing a mechanism to extend the functionality to facilitate end user development. Preliminary user testing suggests that the Euler diagram concept is easily comprehensible for resource categorisation purposes, the concept and application of functions can be understood and applied successfully, and that users perceived the addition of functions increased the usefulness of the application for repetitive tasks.

1 Introduction

User based resource management tasks such as file categorisation are clearly important tasks and yet there are serious limitations within current methodologies to assist users. The traditional hierarchical-based representations are well known to have limitations in the realms of user categorisation, with difficulties occurring when a user wants to categorise resources in more than one place, to change the categorisation structure, or to perform operations on sets of resources that are spread over multiple directories. An alternative methodology is via tagging, and there are methods to handle specific resources such as `mp3`'s or particular types of tagged documents, but they can be problematic if the user wishes to utilise file type based methods for types such as `TeX` or `eps`, for instance; also, tag-based methodologies have the downside that documents with missing or mis-typed tags may be omitted from searches, for example. Some modern operating systems do allow the use of facilities for both tag and file-type based save, search and retrieval methods. As a proposed alternative approach, Euler Diagrams are a means of representing non-hierarchical set-based relationships. They have recently been used in resource management interfaces to facilitate user categorisation, in [15, 5], for instance.

⋆ thanks to EPSRC grant EP/J010898/1.

Looking from another perspective, we know that the organisation of desktop files or links into groups can enhance working memory by utilising the spatial information [4, 12, 14]. However, no methodologies exist, to the best of our knowledge, that bring together the avenues of spatial organisation, categorisation and functional application. Doing so enables the application of functions to spatially organised (or categorised) resources, utilising the user's own spatial memory, whilst also providing the powerful feature of functional application to sets of the spatially organised resources. Therefore, we extend the power of an Euler diagram based interface, enabling the application of a collection of pre-defined functions to collections of categorised resources. The interface enables the user to quickly create a diagram which captures the categories of interest, and their intersections, and utilises a drag and drop facility to apply functionalities to the intersection of categories that are of interest (or to a union of such intersections).

Key ideas. One of the key concepts considered in this paper is the extension of the concept of a *palette*, which is an area where collections of resources of interest can be placed and manipulated, to a *spatial palette* in which these areas are arranged into non-hierarchical regions, as defined by an Euler diagram. We have developed an Euler diagram based interface for user-based categorisation within a non-hierarchical structure, which also allows users to apply functionalities to collections of resources placed within that structure. From an application point of view, we provide means to extend the functionalities in order to encourage end user development and aid future uptake of the interface.

To provide a concrete motivational example, we present one of the user based scenarios that we have developed,set within the `mp3` application domain. Then, in Section 2, we provide Euler diagram interface background information and describe related works. The application is described in detail in Section 3, and some preliminary user testing is reported in Section 4. Subsequent to the user test, we developed a specialisation of the FunEuler interface within the music domain, providing an indication of how the interface can be specialised to cater for domain specific applications; this is reported in Section 5. Conclusions and future work avenues are suggested in Section 6.

User based scenarios. Let us imagine George, a local student DJ, who often creates music based gifts for his friends as well as constructing larger distributions. Related tasks that he performs include: selecting a collection of `mp3` tracks and creating a playlist; naming the playlist and printing out the tracklist as a PDF; zipping a playlist together with a tracklist, or collections of these; emailing the zip file, or uploading it to a music-sharing website; creating multiple zip files of playlists, together with tracklists, plus a global list of the zipped packages in order to distribute more efficiently. By developing a spatial palette (which is a virtual space in which he can arrange his resources and have the facilities to apply the relevant functionalities to them) we will assist George in performing his tasks more efficiently. To gain long term benefits, he would also like to store his music files in an accessible manner, and related tasks he performs include: categorising a collection of `mp3` files, building a structure to hold them, storing (or tagging) the created zip files appropriately. These tasks are also facilitated via the use of an Euler diagram based structure and interface.

2 Euler Diagram interfaces

Venn diagrams [17] are sets of simple closed curves in the plane such that every *zone* (a region that is inside a set of those curves and outside the remaining curves) is a non-empty connected region of the plane. Although there are many variations, especially in terms of *wellformedness condtions* (geometric or topological conditions that one can impose on the diagrams, commonly with the intent of reducing the risks of human mis-comprehension), the main difference is that for Euler diagrams not all of these zones have to be present (i.e. non-empty). Commonly these curves are labelled to indicate the set that they represent, and the diagrammatic systems can be extend to incorporate more information content. Currently there are many application areas for Euler diagram based interfaces related to diagrammatic logic and software specification, and set based data visualisation. Here we present only those interfaces which are directly relevant to our work, being related to enhancing searching and categorisation tasks. The basic architecture and insights into FunEuler were described in [3]. In this paper we specialize FunEuler to a specific context (managing `mp3` files, see Section 5). This provide a simple scenario of use inspired by real life, with motivation easily comprehended by users without explanation.

Related Work. A clickable Venn diagram interface was developed in [13] with the idea of improving web search transparency (the ability to see how sub-queries contribute to the entire result set of a search). It showed the number of results displayed by each term or group of terms in the diagram, but was limited to three search terms and the queries performed upon clicking each region of the diagram were just the conjunction of the terms of the containing curves. Euler diagram based query interfaces have also been used in Traditional Library Environments [11] where the curves represent the query terms, and numbers are used to indicate the number of search term results in the database returned from the Boolean expression associated to the region, but with a slightly different meaning: a region which is outside a curve also means that that curves' search term does not occur in the search results.

Venn diagrams have been used to represent non-hierarchical directories, replacing the traditional hierarchical structure of file systems [15], where diagrams could be drawn with curves representing categories (or tags) and files could be placed within more than one directory by utilising a region of overlap of the contours. In [5], an Euler diagram interface was developed, enabling more general resource management, together with efficient interpretation algorithms to detect the underlying meaning of the regions of the diagram. A reification of an Euler diagram based categorisation structure was integrated with Flickr in [16], utilising the non-hierarchical categorisation structure.

There has been relatively little actual user testing of the Euler diagram concept, with the idea generally being taken for granted as being beneficial. In [2], the comprehension of basic Euler diagrams (without items) that had the same zone sets, but differed in terms of properties such as jaggedness of curves, was examined, whilst in [6], an investigation of the effects of varying wellformedness conditions (e.g. of typical conditions are: no concurrency, no more than two curves crossing at a point, etc.) imposed on user comprehension and preference was performed. In terms of human reasoning processes with Euler diagrams, in [18], eye-tracking experiments to investigate user's focus changes during reasoning tasks were performed. In [7] an in-depth study was presented that aims at measuring the influence of the wellformedness conditions on the comprehension of a diagram; the tests utilised paper based materials.

Background. We view the zones of an Euler diagram as a repository in which to place resources, such as `file`s and `url`s, and so the basic notion of Euler diagrams is extended to capture the placement of items in the diagram; this is similar to unitary alpha Spider diagrams in the diagrammatic logic context [9]. However, in terms of the semantics assigned within this application area, we simply assign the set of tags in a zone's description to that zone, and extend this assignation to any items placed in that zone. Thus an Euler diagram provides a means to build a non-hierarchical categorisation structure, and to use it to categorise resources, as in [5].

3 Application description

We describe the FunEuler interface, together with justifications for the design decisions adopted. In Figure 1 a screenshot of the new FunEuler application developed is shown. The application is available at [1]. The main portion of the window real estate contains the diagram itself and this is where the main user-interaction occurs. At the top of the left hand pane, a list of icons representing the possible operations that the user may apply (to regions of the diagram) is displayed. Below this operations list there is a special area displaying a simple diagram which indicates the *Results* set. This essentially depicts the output from the application of the user-selected operations (see Subsection 3.2 for more details). At the top of the interface, a standard menu bar enables the user to select different interaction modes: modify a curve (Arrow icon), draw new curve (Pencil icon), query the diagram (Eyedropper icon), reset the zoom level (Magnifying glass icon), tag selected items (Tag icon) and save the diagram (Camera icon).

3.1 Diagram construction and interaction

We allow users to construct a diagram by adding curves, in the form of ellipses, one at a time by a simple mechanism of left click and drag to specify one axis whilst using the mouse scroll wheel to specify the other axis (cf. Figure 2 top-left). This allows fast user construction of the diagrams, as well as a fast interpretation of the zones of the diagram [5]. The downside is that it slightly restricts the freedom of users who may wish to draw other types of curves, but the use of more general curves can be considered as a future extension where the associated trade-offs will be investigated. However, we emphasise the important point that in this application domain users only need to construct diagrams which have a superset of their required collection of set intersections that they need to represent and so concerns relating to the complexity of diagram construction [8] are reduced.

At the moment of creation of a curve, the system automatically assigns a random colour to the curves, as well as a choice of set name (according to a predetermined sequence), but these names and colours can be then edited by the user. This facilitates

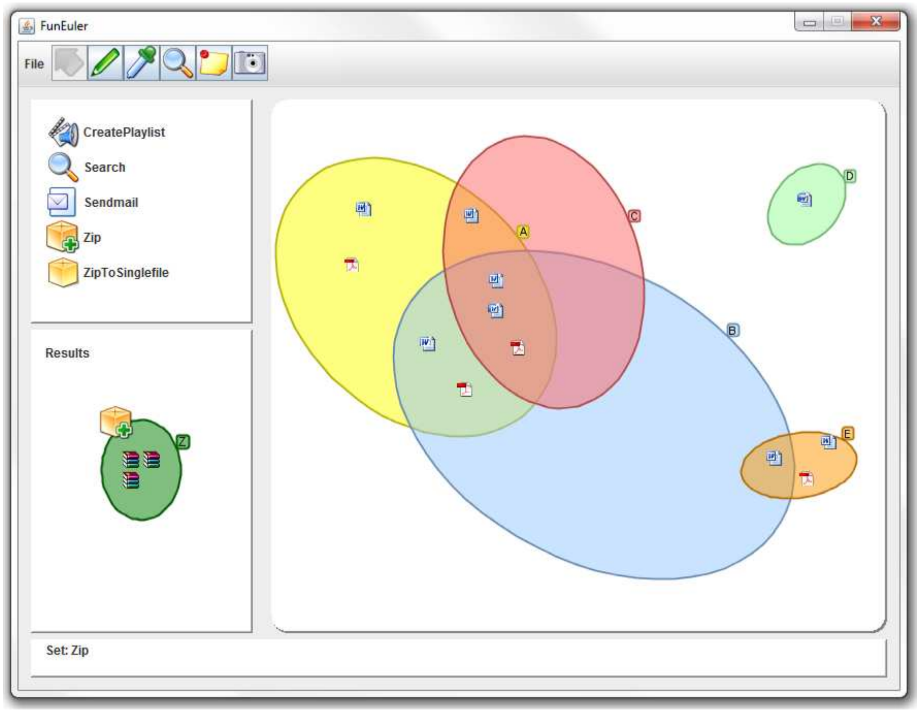


Fig. 1. A screenshot of the FunEuler interface.

rapid user construction of diagrams via ellipse addition without the interruption of naming and choosing a colour for each curve, which could interrupt the flow of creation. The collection of parameters associated to each ellipse are easily accessible through a pop-up menu via a right-click action (cf. Figure 2 top-middle). Existing ellipses can be quickly modified by translation, rotation or by altering the size of its axes.

Although the general advice would be that users constructing categorisation diagrams should keep them as simple as possible, human readers of diagrams may have difficulties identifying zones correctly if complex diagrams which have small zones with a lot of bounding curves, or if zones are complex spatial regions (e.g. in the general case they can be disconnected or non-simple regions) are constructed. Therefore, we assist in helping a reader to understand a categorisation by providing the possibility of querying a zone: by selecting a point p within any zone using the eyedropper the entire zone containing p is highlighted, and the zone descriptor displayed in the status bar (cf. Figure 2 top-right).

To aid user navigation within larger diagrams, we allow the usual scrollbars for horizontal and vertical directions, but we also provide spatial zooming: this is accessed via a select zoom option and mouse wheel scroll in and out, providing a larger working area when needed. This facility is likely to be useful in cases when users are working with a lot of categories, or when there are a large number of items in an intersection of categories and one needs to enlarge the working space (i.e. to enlarge the region that is the spatial palette under consideration).

Item management. We utilise a drag and drop protocol for resource classification, simply placing items within the relevant region of the diagram. Item properties are visualized through tool tip text while the status bar, at the bottom of the interface, provides its current categorization (cf. Figure 2 bottom-left). Each item can be interactively repositioned and consequently re-classified. In order to not restrict the selection (and hence the functional application) of items to those sets of items corresponding to entire zones we also allow the selection of multiple items via a common box-selection (cf. Figure 2 bottom-middle/right).

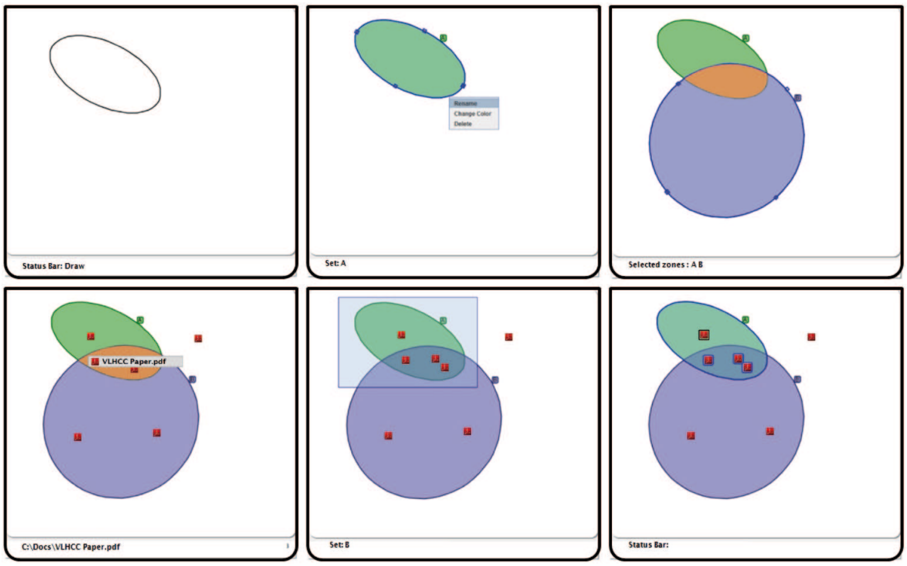


Fig. 2. A sequence of screenshots demonstrating FunEuler basic functionalities.

3.2 Operational functionality

At the core of the FunEuler functionalities is the ability to select and apply one, or more, *operations* to regions of the diagram. Performing operations on zones uses the familiar drag and drop: the user simply drags the operation icon onto the desired zone of the diagram. To perform operations on multiple zones (or similarly for other selected sets of items), the user must first perform a selection (by using the eyedropper) on all of the zones that he/she wishes to apply the operation to, and then drag the operation icon onto one of the selected zones.

To demonstrate the principal of functional application within the Euler diagram structure we developed several operations including: zip (which individually zips all files in a region, creating a collection of zip files), ZipToSingle (which zips all files in a region to a single zip file), createplaylist (which creates a playlist when applied to a collection of `mp3` files) and search (described below).

Tagging and Searching. The Euler diagram based interfaces commonly have a means to export to a tree based file system using conjunction of categories as node labels. However, we provide some specialised tag-based support for this interface. In general item tagging can be performed by adding information “inside” the item (e.g. utilising the genre field for `mp3` files or tag data for office documents) or “outside” by utilising an item tagging database.

Now, if diagrams contain a large number of items then the user’s ability to find and use specific items may be adversely affected. Therefore we offer the user the option of hiding tagged items, so that a user can hide items classified within the zone, thereby controlling what is actually visualised. Then, users can display tagged items by using a specific Search functionality, which displays all of the items whose tags correspond to the appropriate intersection of categories. We have developed a simple yet powerful Search functionality that relies on the Windows Search feature (which was introduced with Windows Vista/Seven) which allows the user to query, using an SQL-like language, the `SistemIndex` catalogue of Windows machines; since we offer support for this kind of operation we have access to functions for searching and modifying tags.

Developing new functions. FunEuler is designed to let the user to program to extend its functionalities. The very essence of the operational application within the interface is allowing users to assemble operations and this could be viewed as building a simple program or macro to support their tasks. The interface allows programmers to develop new operations by using their favourite scripting language (e.g. php, python, etc). Indeed, FunEuler provides access to two lists for the items and the zones that the operation is to be applied to: one list containing the pathnames of the items and the zones they belong to, and another list reporting the zones and the items that each zone contains. Each operation can produce zero, one or several files as result. As an example, the `zip file` operation produces a zipped file for each pathname in the input list, whilst the `zip to single file` operation produces one single `zip` file containing all of the files of the input list. On the other hand, the `send by email` operation does not produce any result files. Independent of the number of files produced the Results pane supports their representation in a special single-curve ED, where file icons are automatically placed within the curve representing the results set.

Closure. The application of an operation to a region of a diagram causes the generation of a new diagram in the Results part of the application window, and since the Results set is a diagram in its own right, operations can be applied to it generating a new Results diagram arising from the sequential application of the selected operations. Thus we can view the class of diagrams in FunEuler as being *closed under the application of operations*. Figure 3 shows the operations’ icons within the Results set; in the left hand side the items are the result of one operation (create playlist on `mp3`s) whilst on the right they are the result of two operations (create playlist followed by zip). Subsequent operation icons are added consecutively, above the previous operations’ icon. Due to the closure we allow the user to drag the Results set onto the main diagram pane. This permits the user to categorise items which are the result of one of more operations alongside the original items that were already displayed in their diagram.

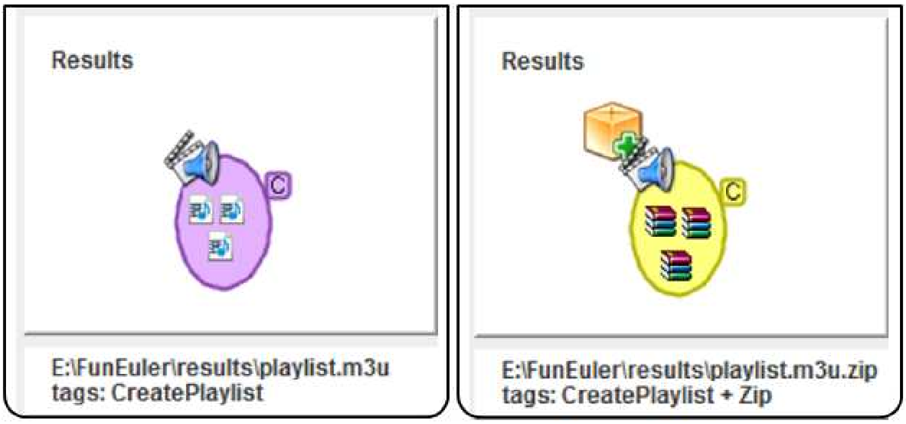

Results
C
E:\FunEuler\results\playlist.m3u
tags: CreatePlaylist
Results
C
E:\FunEuler\results\playlist.m3u.zip
tags: CreatePlaylist + Zip

Fig. 3. Two screenshots showing the results of operations: (left) A single operation, *Create Playlist*; (right) Two sequential operations, *Create Playlist* and *Zip*.

3.3 Comparison with other tools

We decided not to try to compare the Euler diagram concept directly with other existing methodologies, because it is difficult to determine a suitable comparator and it also seems likely that tasks set would have a good chance of being biased towards one methodology. For example, a comparison of the Euler diagram concept for categorisation/resource management versus a hierarchical structure is likely to favour the Euler diagram concept for tasks that utilise overlapping categories, providing the underlying concept of Euler diagrams is not challenging for users. Furthermore, knowledge of users' level of comprehension and abilities with this underlying concept is a necessary precursor to allow us to accurately examine their abilities with the concept of the application of functions to regions of the diagram.

4 User testing

We decided to test users' conceptual understanding of the basic concept of Euler diagrams within this domain, together with their perception of its utility. Since we also wish to know if users can understand and apply functions, we also perform a basic test of this, keeping the tasks fairly straightforward in order to test the basic concept, with significantly more complex operational tasks being planned for future studies (as well as a more direct comparison within a more specific domain application that we have subsequently developed and is described in Section 5).

We test the concepts within the FunEuler tool we developed, which means we cannot really separate the conceptual effects of the actual representation from the effects of the particular tool, since the tool may restrict a user's actions and thus alter their behaviour; to acknowledge this, at the end of the tests, we asked users if they were aware of anything they would have liked to have been able to do but could not, either because of the representation or because of the tool. However, we believe these effect of the tool on the judgement of the representation will be minor in this instance due to the simplicity of the notation and the interface, whilst the tool based approach brings many benefits in that we can trace with the behaviour of the users in detail (e.g. the number of clicks, the number of corrections made during the test, etc. . .).

We describe our experimental design, some of the actual tasks set to the users, and a summary of the results of the user tests. We provide very high-level broad hypotheses below, but note that we are actually testing a restricted class of fairly small simple diagrams (no more than five curves or twelve zones), and as always any statement of results must really be restricted to the diagrams used and the tasks set within the experiment; we take this restriction for granted in the following. However, we are provided with an indication of the potential usefulness of the interface.

Hypotheses.

H1 Users can easily comprehend, create and modify Euler diagram based categorisations using FunEuler.
H2 Users are able to utilise the application of functions using FunEuler effectively.
H3 Users perceive FunEuler as an application they would find useful to improve their efficiency of operations.

The intention of the user testing is to investigate Hypotheses $H1$ and $H2$ using quantitative methods, by the recording of correctness of task completion, and efficiency in terms of timing, whilst we investigate the subjective questions related to Hypotheses $H3$ using qualitative methods. In order to test these hypothesis we carried out two different test sessions: test session 1 focused on $H1$ and $H3$, whilst test session 2 focussed on $H2$ and $H3$.

Participants. We recruited 19 volunteers, twelve of whom were male, aged between 21 and 31, who were a mix of undergraduate Computer Science students and Ph.D. students at the ISISLab, Università di Salerno. Three of these participated in pilot testing of successive versions of the experimental materials (e.g. the tasks and questionnaire). This allowed us to thoroughly check the experimental set-up, altering the language and providing more clarification of the tasks where necessary. The remaining sixteen participants were equally divided between two tests sessions (each of which were of approximately 25 minutes duration).

Test setting. In terms of the test conditions, both tests were carried out in a laboratory setting where each user had a dedicated laptop. Each test was explained orally and guided via a printed handout offering a short description of each of the stages. At the end of each step the user was asked to take a *snapshot* of the diagram, by clicking a special button in the application, in order to record both the duration of the last task and the current state of the diagram. At the end of each session, the users were asked to fill in a questionnaire, consisting of 26 questions divided into 6 sections: general application, set creation, set manipulation, item categorisation, executing operations and perceived effectiveness of FunEuler in accelerating daily activities. These were measured on a Likert scale with score range $[-3, 3]$.

Experimental design. In the first test session each user was provided with a sequence of five stages, with each stage describing a basic concept or functionality of FunEuler. A task was presented to the user which required the utilisation of that functionality in order to modify the diagram, for instance. Thus we could check the users' comprehension of the induced modifications. Sample tasks included: reproduce a given diagram presented on paper utilising the tool, and to count the number of zones into which a given set was split. For each stage we recorded the completion time and we measured the percentage of the tasks performed correctly (e.g. the number of sets that were correctly represented, or the number of zone sets that were correctly counted). The second test session was aimed at assessing users' ease of comprehension and their application of operations to zones containing items. This test session consisted of three stages. The first stage was to perform item tagging in the similar manner to that of the first test, whilst the next two stages focussed on the application of a zip operation to certain zones of the diagram. We provide a brief description of the tasks below.

User Tasks. We consider user tasks divided into the following categories: Interpretation (given a diagram interpret its meaning), Development (given some categories and items construct a diagram) and Functional (utilise diagrams, applying functions). Note that, as mentioned earlier, we had decided it was a priority to test the fundamental aspects of the concept and the application and to leave the complex task categories like end-user development to a future date.

Table 1 show the basic results related to the tasks that comprised the two test sessions. We briefly describe the tasks and the criteria that we adopted to evaluate their execution, where necessary:

Diagram Construction To create a diagram containing certain specified zones. The presence of the given zones is checked.

Diagram Modification To modify a diagram by altering the sets represented (i.e modifying the curves which changes the set of zones present).

Zone Counting To count the number of zones contained belonging to certain set (i.e contained within a certain region).

Diagram Reproduction To reproduce a printed diagram. The presence of the given zones is checked.

Item Tagging To place eleven files onto zones of a given diagram. Each file is named according to the zone it should be placed in. The reason for this was to mimic the pre-existing knowledge of file content that a user would normally have at the time at which he/she is about to categorise a file. The correctness of the placement is checked.

Zip Items in one zone To apply the zip operation to files belonging to one zone which represented the intersection of two sets.

Zip Items in multiple zones To apply the zip operation to files belonging to multiple zones from a single set.

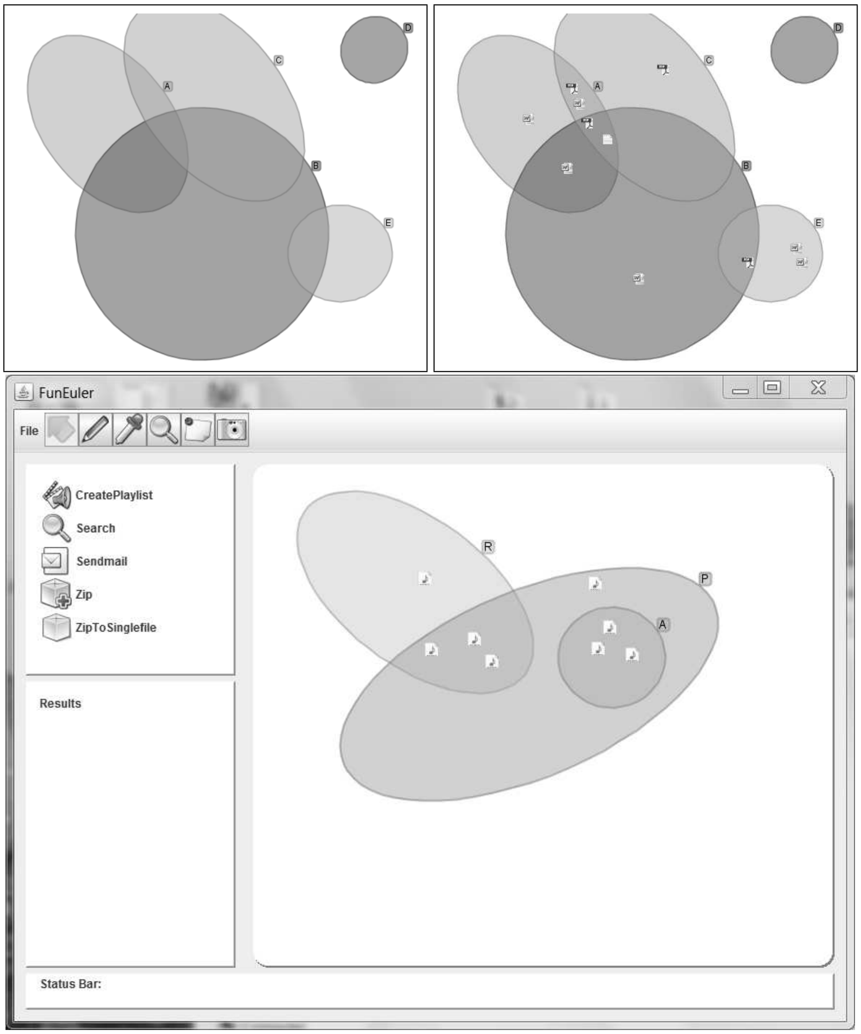
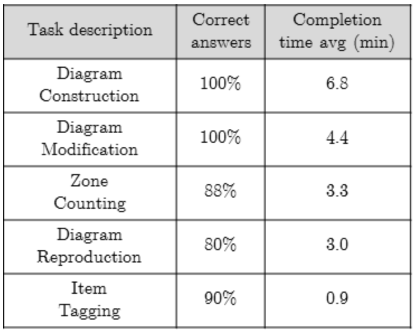


Fig. 4. Screenshots from one of the tests

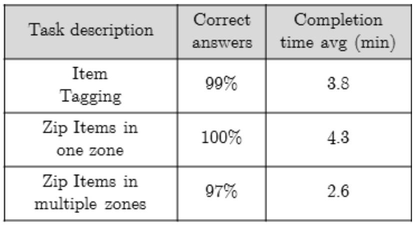
In Figure 4 we present screenshots of diagrams constructed by users of the test. In the upper part the user has reproduced the diagram in Figure 1: on the left the diagram has no items, whilst on the right she had categorized 12 files, as requested, and a single file was misplaced (placed in E instead of D). In the lower part of the same figure we show a diagram used to categorize `mp3` files.

4.1 Results

In Table 1 we present a summary of the results of the two test session that were carried out. Within the context of the diagrams and tasks used, Hypothesis $H1$ is supported by a high score on the tasks of Diagram Construction, Diagram Modification and Diagram Reproduction. Hypothesis $H2$ has been tested in tasks Zip Items in one zone and Zip items in multiple zones which also report $\approx 100\%$ success rates.



Task description	Correct answers	Completion time avg (min)
Diagram Construction	100%	6.8
Diagram Modification	100%	4.4
Zone Counting	88%	3.3
Diagram Reproduction	80%	3.0
Item Tagging	90%	0.9



Task description	Correct answers	Completion time avg (min)
Item Tagging	99%	3.8
Zip Items in one zone	100%	4.3
Zip Items in multiple zones	97%	2.6

Table 1. Tests results: (left) first test; (right) second test.

By comparing the two tests session results we could evaluate how the use of operations influenced the users' perceived usefulness of FunEuler: the average score was 0.13 for participants of the first test session but 1.88 for participants of the second test session (the score range is $[-3, 3]$). A one-way ANOVA was calculated on participants' ratings of the question "is FunEuler useful for repetitive tasks ?". The analysis was significant, with $F_{1,14} = 5.0,\ p = 0.04$.

The accurate responses to the tasks indicate that the underlying conceptual use of Euler diagrams for categorisation was easily grasped by users. The functional application utilising this structure did not appear to be a large conceptual step for them. The perceived usefulness of the FunEuler application was significantly enhanced by the utility for functional application. Users also reported that they would use the application in the future, suggesting areas where it would be useful, and some even stated that they would be interested in developing the associated functionality themselves.

5 FunEuler specialisation with `mp3` files

FunEuler can be viewed as a flexible framework on which to host services based upon an Euler Diagram representation of information. As an example of this flexibility, and as an extension of FunEuler following the experiment, we developed a prototypical `mp3` files management system. Since the music domain is of interest to the student participants, and since there are applications available to deal with `mp3` files, we focussed on this specialisation area.

The diffusion of digital music has radically changed the manner of listening to music for the average user, and one fundamental step toward this evolution is the widespread adoption of the `mp3` file format [10]. `mp3` files are commonly created by the so-called *ripping* of CDs: CD audio tracks are digitally recorded by using a computer and compressing them into an `mp3` file. This process is straightforward and a number of applications can perform it as well as attempt to fill the ID3 fields (identification for `mp3`) by accessing online archives (e.g. `http://www.gracenote.com`). A second, and perhaps the most relevant, method to obtain `mp3` files is by buying them from online music stores (e.g. iTunes, Amazon), which usually provide correctly filled in ID3 fields. Despite these two widespread methods to obtain music, there is a large number of `mp3` files that are not tagged at all: we sampled more than 8000 `mp3` files available on our students' accounts, by asking them to run a small application that calculates a statistic on the content of the field Genre in each `mp3`, and we found that around 75% of them had empty or bad (containing random characters) Genre tags. Thus we inferred that the Genre field would be a suitable testbed for FunEuler tagging capabilities.

New functionalities. We implemented the operations of searching `mp3` files by genre and tagging `mp3` files. By combining the use of these operations it is possible to categorise large archives of `mp3` files using few mouse clicks. The idea is to implement a cycle of *search-categorise-tag*: search for untagged files, categorise them by dragging and dropping their icons onto the zone representing the correct genre, and tag files by applying the tagging operation. The search for untagged `mp3` files can be obtained by applying the Search operation to the “Universe” zone, and in this specialisation, the number of untagged `mp3` files is displayed within the zone; clicking on the number causes a window containing the list of files to be opened. It is now possible to drag and drop the `mp3` file icons onto the desired zone of the diagram in order to categorise them. The tagging operation assigns the set of tags associated with the zone containing a file to the Genre ID3 field.

6 Conclusion & Future works

We have brought together concepts of visual non-hierarchical classification, spatial arrangements and functional application in the development of a novel Euler diagram based interface called FunEuler. The prototype application developed was utilised in user testing, which investigated if the Euler diagram concept was easily comprehensible for resource categorisation purposes, if the application of functions could be performed successfully, and if users perceived this idea to be useful either in its current state or together with future functionalities. The results were extremely encouraging.

The FunEuler interface also provides a firm base for future developments in many directions. For example, we could extend the power of the labels of curves, allowing a mixture of user defined categories (or tags) such as “Computational Geometry” and pre-determined types such as TeX files, or to allow functional labels such as “Before this year”, which would filter for files based on the timestamp. Allowing any Boolean expressions on labels would assist with scalability of the diagrams since more information could be encapsulated in the labels instead of via the addition of more curves. A useful option could be to allow users to select from a set of curves in a library (e.g. files modified yesterday, PDF files, etc). In terms of repetitive tasks, we observe that choosing specific groups of contacts for sending email to is a task that a user often needs to perform several times a day, and so enabling a fast mechanism to realise group selection as well as to facilitate repeated usage of related group selections could be of great assistance to end users, and we will extend an Euler diagram interface for such tasks.

Furthermore, there are many potential application areas within which the developed concept and interface could be applied. For instance, we briefly reported on the specialisation for `mp3` files and we plan to run future user testing within this domain. Secondly, in the workflow application area (e.g. see Alfresco), we could allow the drag and drop of a `TeX` document into a particular region, automate the running of macros such as PDFTeX and observe the results appear automatically elsewhere as an item in the PDF category. Users could then understand and track workflow involving multiple people on different systems. Thirdly, we will consider Euler Diagram Programming, where we will assign behaviour to curves instead of to the zones. For example one might have "Hotel" and "Food" categories with different "print" functions but their intersection might make use of both functionalities, although one must deal with the usual multiple inheritance problems, of course (e.g. we do not want to print off a date twice due to its appearance in each categories' behaviour).

One weakness of the representation is the need for continuous navigation in order to find specific sets/regions, particularly in larger diagrams. Whilst the representation may be an effective way of leveraging the spatial memory of the user, performance of actions by hand without specific automated support from the application may be annoying. A deeper investigation into the advantages and disadvantages of the representation, and exploration of potential tasks and application areas is a next step.

A wider investigation of the usability of EDs in a general context is required. There are many unanswered questions, such as: how large (in terms of the number of zones and number of curves) can a diagram be to be effectively manipulated by an average user; how well does the representation scale with the increasing of the complexity of the diagram; what effect does the representation have on user behaviour? Such questions require specifically targeted testing.

Acknowledgments. We wish to thank the anonymous reviewers for their useful comments, particularly reviewer number 3 whose insightful comments will be invaluable in future works.

References

1. Funeuler. `http://www.isislab.it/projects/FunEuler/`.
2. F. Benoy and P. Rodgers. Evaluating the comprehension of euler diagrams. In *Proceedings of IV 2007*, pages 771–780, 2007.
3. Paolo Bottoni, Gennaro Cordasco, Rosario De Chiara, Andrew Fish, and Vittorio Scarano. Personalised resource categorisation using euler diagrams. In Maria Costabile, Yvonne Dittrich, Gerhard Fischer, and Antonio Piccinno, editors, *End-User Development*, volume 6654 of *Lecture Notes in Computer Science*, pages 251–257. Springer Berlin / Heidelberg, 2011.

4. Andy Cockburn. Revisiting 2d vs 3d implications on spatial memory. In *AUIC ’04: Proceedings of the fifth conference on Australasian user interface*, pages 25–31, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.

5. G. Cordasco, R.De Chiara, and A. Fish. Interactive visual classification with euler diagrams. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing VL/HCC 2009*, pages 185–192. IEEE Press, 2009.

6. A Fish, B. Khazaei, and C. Roast. Exploring human factors in formal diagram usage. In *Proceedings of Engineering Interactive Systems conference 2007*, LNCS 4940, pages 432–448, 2008.

7. Andrew Fish, Babak Khazaei, and Chris Roast. User-comprehension of euler diagrams. *Journal of Visual Languages & Computing*, 22(5):340 – 354, 2011.

8. J. Flower, A. Fish, and J. Howse. Euler diagram generation. *Journal of Visual Languages and Computing*, 19:675–694, 2008.

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

10. ISO/IEC 11172-3:1994. *Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 3: Audio*. ISO, Geneva, Switzerland.

11. A. Verroust-Blondet. J. Thièvre, M. Viaud. Using euler diagrams in traditional library environments. In *Proceedings of Euler Diagrams 2004*, pages 189–202. Electronic Notes in Theoretical Computer Science, 2005.

12. Akrivi Katifori, George Lepouras, Alan Dix, and Azrina Kamaruddin. Evaluating the significance of the desktop area in everyday computer use. *International Conference on Advances in Computer-Human Interaction*, pages 31–38, 2008.

13. L. Langer and E. Frokjaer. Improving web search transparency by using a venn diagram interface. In *Proceedings of NordiCHI 2008*, pages 249–256. ACM Press, 2008.

14. Maarten, Mary P. Czerwinski, Maarten Van Dantzich, George Robertson, and Hunter Hoffman. The contribution of thumbnail image, mouse-over text and spatial location memory to web page retrieval in 3d. pages 163–170. Press, 1999.

15. R.De Chiara, U. Erra, and V. Scarano. VennFS: a Venn Diagram File Manager. In *Proc. of the Seventh International Conference on Information Visualisation, IV 2003, 16-18 July 2003, London, UK*, pages 120–126. IEEE Computer Society Press, 2003.

16. R.De Chiara, A. Fish, and S. Ruocco. Eulr: a novel resource tagging facility integrated with Flickr. In *Proceedings of the AVI 2008 Advanced Visual Interfaces Conference*, pages 326–330. ACM Press, 2008.

17. F. Ruskey. A survey of Venn diagrams. *Electronic Journal of Combinatorics*, 2005. www.combinatorics.org/Surveys/ds5/VennEJC.html.

18. A. Shimojima and Y. Katagiri. An eye-tracking study of exploitations of spatial constraints in diagrammatic reasoning. In *Diagrams 2008*, LNAI, pages 74–88, 2008.

Multi-Attribute Glyphs on Venn and Euler Diagrams to Represent Data and Aid Visual Decoding

Richard Brath

Oculus Info Inc., Toronto, ON, Canada
richard.brath@oculusinfo.com

Abstract. Representing quantities on Venn and Euler diagrams can be achieved through the use of multi-attribute glyphs. These glyphs can also act as an aid to assist in the visual decoding of the membership of segments within the diagrams and convey other data attributes as well.

1 Overview

Instead of area-proportional Venn and Euler diagrams to indicate quantities, this approach uses separate overlaid glyphs to decouple the representation of data from logical combinations. It also uses glyph attributes to assist in visual decoding of membership of regions. This approach can scale to higher-order logical diagrams and potentially offer more accurate visual estimation than area-proportional techniques.

The depiction of data on set diagrams is useful in various applications (e.g. Boolean queries, genetic informatics). Area-proportional set diagrams have become popular in research (e.g. [1-6]) and software (e.g. see eulerdiagrams.org). However, the area-proportional approach has shortcomings, such as:

1. **Visual comparison of irregular areas is difficult**. Information visualization researchers indicate difficulty with visual comparison of areas and/or a preference for using length instead of area for faster visual comparison (e.g. [7,8,9,15]). In our casual test, only 8% (2 out of 25 people) correctly identified the region of different area on a 2 way Venn as opposed to 80% correctly identifying the circle of different area out of three circles, each test having one item of 20% different area.
2. **Area accuracy vs. aesthetic shapes**. Researchers prefer circles and other aesthetic shapes[6], but the areas (particularly circles) may have a degree of error, typically increasing with higher order sets. e.g.[1]. Wilkinson [6] says “Higher-order Venn diagrams can be drawn on the plane with nonconvex polygons, but they are difficult to compute for more than a few sets and are difficult to decode visually.”
3. **Negative values**: Areas cannot represent negative values unless coupled with another visual attribute, such as hue (e.g. red/green) or shape (e.g. arrows).

Also, discussions with prospective users revealed concern for visual decoding of set membership for a region in complex diagrams, such as higher-order Venn diagrams.

3rd International Workshop on Euler Diagrams, July 2, 2012, Canterbury, UK.

Instead, a glyph-based approach is considered. The use of glyphs within set diagrams is not new. Glyphs have been used to represent items in a dataset (first 3 in fig. 1). Spoerri's approach [10] reduces each region of a Venn diagram to a glyph, each glyph indicating the particular Boolean combination by its relative position and shape.

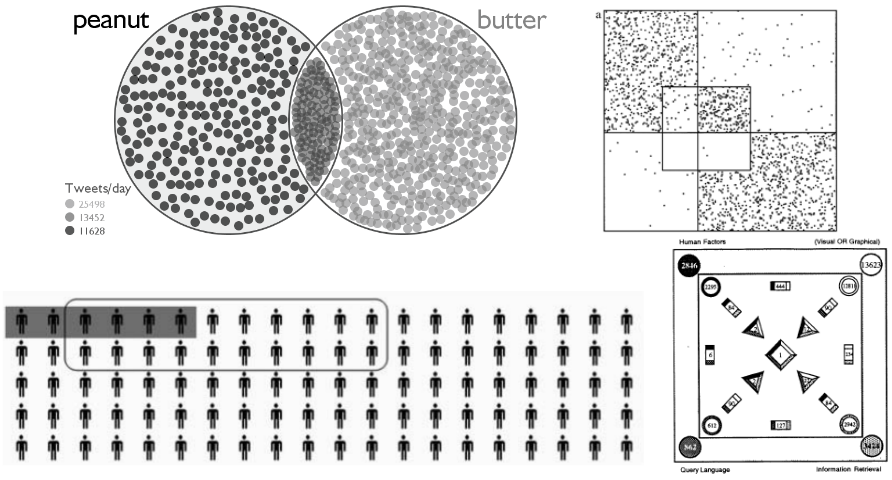


Fig. 1. Top Left: TwitterVenn [11] uses simple glyphs per tweet matching each of three search terms. Top Right: Edwards' Carroll diagram [2] uses both proportional areas (to indicate expectations) with dots (to indicate observations) to draw attention to regions with unexpected results. Bottom Left: [12] depicts a uniform density of glyphs with membership indicated by shading or outlined boxes. Bottom Right: InfoCrystal [10] reduces each region of a Venn diagram to a glyph, where shape and position indicate the Boolean relationship, and the number indicates the count of items within each region.

The contribution of this paper explores, in section 2, the use of glyphs (pictographic and scaled glyphs) to indicate quantities and the use of additional visual attributes to indicate set membership or other data. Results are discussed in section 3.

2 Glyph-based Approach

Our approach is focused on the use of glyphs to decouple the depiction of logical relationships (e.g. Venn and Euler diagrams) separate from the depiction of quantities. By decoupling the quantity from the set diagram, visual attributes more amenable to fast estimation (e.g. size, origentation and color) can be used [7-9]. This approach enables the use simple aesthetically pleasing diagrams of set representations to show the logical relationships between the sets; while using separate glyph(s) within each region to indicate a) quantity of items within a given region, b) indicate set membership to aid visual decoding and c) potential additional attributes.

2.1 Sketches and Real-Data Mockups

To quickly iterate through conceptual ideas, loose sketches were followed by mock-ups using simple sets of real-data. Loose sketches can reveal limitations of promising ideas when implemented with real-data, (e.g. occlusion, imperceptible differences, large dynamic ranges, etc). For rapid mock-ups, we divided the Titanic passenger list into 4 sets for a Venn diagram and 3 sets for an Euler diagram which resulted in useful properties such as empty segments, small segments and large segments.

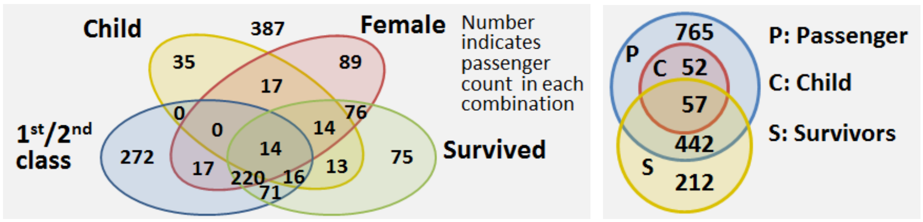


Fig. 2. Left: Venn diagram of Titanic passengers by 4 attributes showing passenger counts. Right: Euler diagram of Titanic passengers and crew.

2.2 Unit Markers and Pictographs

Markers of a fixed size can be repeated to represent quantities ranging from simple dots to pictographs e.g. Isotype [13]. Additional data can be represented on each marker, e.g. using color or sub-shapes. For example, “Social Stratification in the United States” [14] uses pictographic markers with human figures indicating five variables, through 1) background color (occupation), 2) shape (gender), 3) pairing (marital status), 4) extra outline (dependents), and 5) figure color (race). We have also used this approach successfully, e.g. [15].

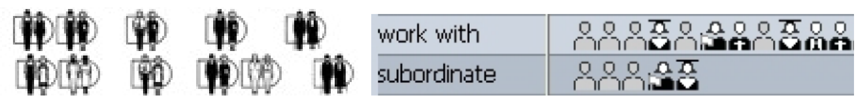


Fig. 3. Pictographic glyphs indicating multiple data variables, from [13,14].

However, a pictograph approach has some challenges:

- Some regions of the set diagram are too small to fit the pictographs. The addition of the leader line could increase the effort to visually decode the relationships.
- The irregular shape of some regions of the set diagram requires an irregular placement. Pictographs organized linearly can be visually estimated by length, which is preferred to visual estimation of area (e.g. [8]).

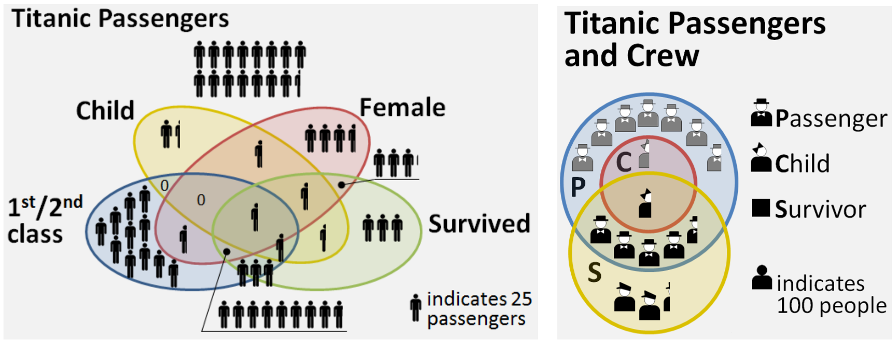


Fig. 4. Pictographs on set diagrams. Leader lines are used when pictographs do not fit regions.

2.3 Scalable Glyphs

Scalable glyphs are a single glyph for each region, sized by the quantity of items associated with that region. Simple glyphs, such as bars varying in length or circles varying in radius, can effectively convey quantities [8,9,16]:

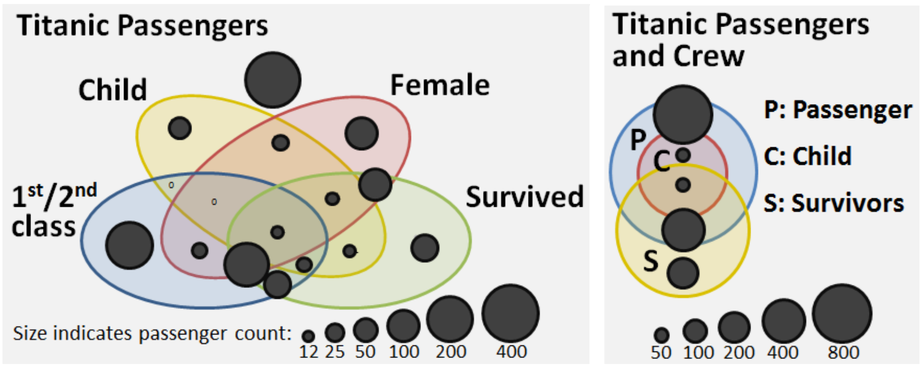


Fig. 5. Scaled glyphs indicating quantities on both Venn and Euler diagrams.

2.4 Indication of Set Membership

With higher order set diagrams, it can be more difficult to perceptually decode the membership for a given component of interest [6]. There are many possible approaches to indicate set membership using either the set diagram or the glyphs.

Background Color: Color can be used, but is problematic. It is challenging to decode the color in intersections as color is not understood as separable [16].

Background Texture: Textures have been used to aid in identification of set membership e.g. [17, 18]. Distinguishing regions by a heterogeneous channel-based approach [19] could be more effective. However, in small regions, textures may not be clearly distinguishable or the glyphs may occlude textures.

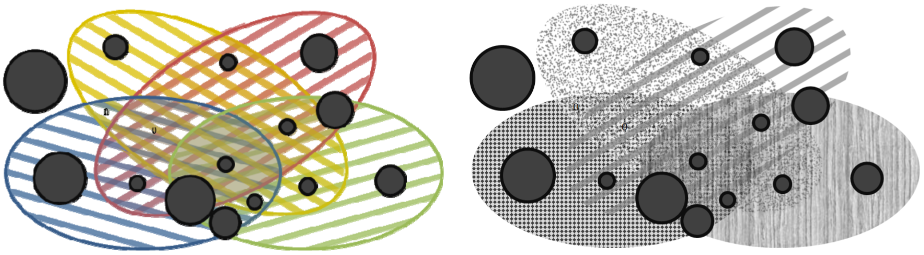


Fig. 6. Background textures can assist in decoding membership.

Glyph with Colors: The same coloring used in the set diagram can be reused in glyphs to indicate membership. Rather than blend colors, however, the colors can be kept separate within the glyph. The layout of the color could be organized as stripes, or radially resembling a bullseye or pie.

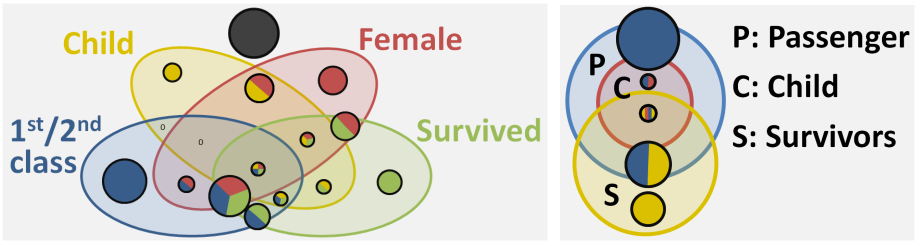


Fig. 7. Colored glyphs use same colors as the sets to aid in identification of set membership.

Glyph with Oriented Whisker: In some set diagrams (e.g. Venn) the placement of the label is typically around the perimeter which can be leveraged by the glyph, specifically by modifying the shape with an added whisker [20] oriented along the same vector from the center of the diagram to the label. To visually decode the membership of any bubble, the viewer can read the orientation of the whisker, similar to decoding the hands of a clock or the spokes of a wind rose.

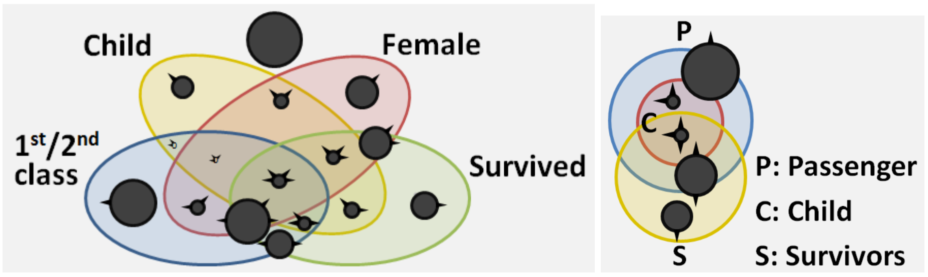


Fig. 8. Glyphs with added whiskers oriented based on set memberships.

The glyph-based approach also allows for additional visual attributes to convey additional data values. For example, the whisker-based glyphs can use:

- Traditional visualization attributes, such as brightness, hue, texture
- Shape-based attributes, such as closure, curvature or edge type
- Each whisker-shape can be independently modified to indicate a data attribute with respect to the set membership, for example whisker length or width
- The internal area of the glyph can be used, in larger glyphs, for example, as a pie chart or with a pictograph.

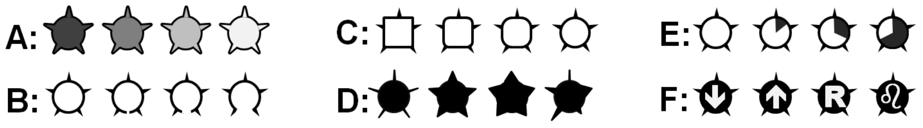


Fig. 9. Example whisker glyphs with additional visual attributes, a) brightness, b) closure, c) curvature, d) whisker width and length, e) internal pie, f) internal pictograph. Negative values could be connotatively conveyed by fill (e.g. red/green hue) or pictograph (up/down arrow).

The whisker-based approach may work well with Venn diagrams, but has may have issues with Euler diagrams and issues where whiskers are potentially occluded. The image below shows a 5-way Venn diagram that has been modified from ellipses to increase the size of the smaller regions to make the technique more workable.

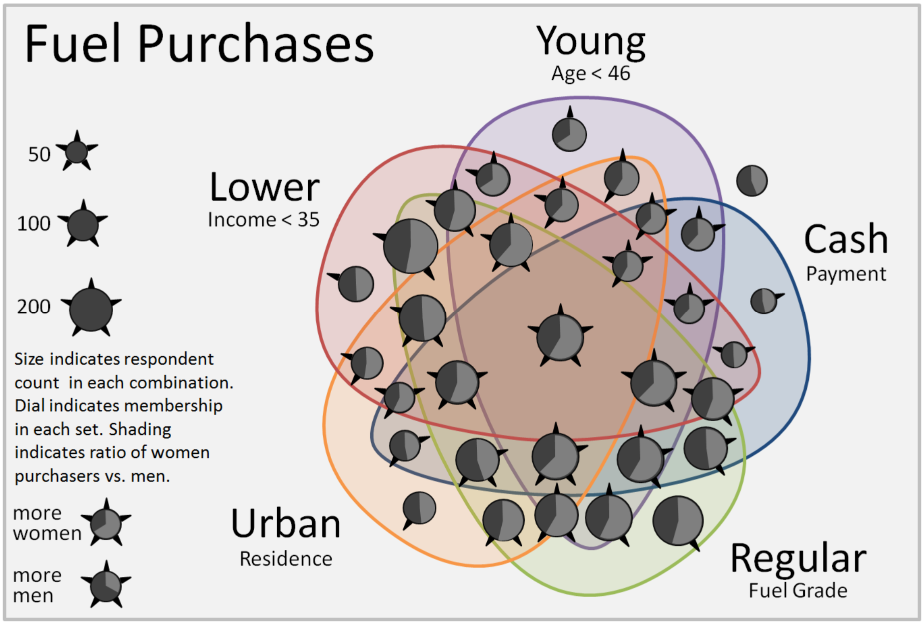


Fig. 10. Whisker glyphs on a 5 way Venn diagram showing data from a survey of 5000 people purchasing fuel, in sets by age, payment type, fuel grade, residence and income. Size of glyph indicates number of respondents in a given component, whiskers indicate set membership, and angular shading indicates the ratio of female to male purchasers of a component.

3 Discussion and Next Steps

Our contribution shows that glyphs can be used to separate the representation of data such as quantities from the representation of sets. Glyphs can:

- Indicate data layered over set diagrams, either as scalable glyphs or as pictographs, and simple size is preferred for fast visual estimation as opposed to irregularly shaped areas [16].
- Also indicate additional data attributes, such as set membership or other data attributes, using visual attributes such as color or orientation of sub-shapes.

While current research in visual comparison indicates this approach may work, evaluation is required to validate. The examples provided indicate various limitations:

- **Glyph size** needs to be carefully managed. Glyphs too big can result in occlusion or require an offset and leader lines. Glyphs too small can be difficult to add additional visual attributes, e.g. for visual decoding.
- **Glyph colors** can effectively represent set membership except color can be difficult to discern when used internally on small glyphs.
- **Glyph whiskers** can effectively represent set membership when set labels are organized around the perimeter, such as in Venn diagrams; but is problematic when sets are distributed throughout the plane.

Implementation on a wide variety of data sets and testing with users requires further effort. Other work could include 3D and interaction techniques, for example, on interaction, extend whiskers to set labels to aid interpretation of whiskers.

References

1. Chow, S., Ruskey, F.: Drawing Area-Proportional Venn and Euler Diagrams. Lecture Notes in Computer Science 2912:466–77 (2004)
2. Edwards, A.W.F., Edwards, J.H.: Metrical Venn diagrams, Annals of Human Genetics 56: 71-75 (1992)
3. Micallef, L., Rodgers, P.: eulerAPE: Drawing Area-Proportional Euler and Venn Diagrams using Ellipses. EMEA Google Scholars Retreat 2011 www.cs.kent.ac.uk/pubs/2011/3119 (2011)
4. Chow, S., Rodgers, P.: Constructing Area-Proportional Venn and Euler Diagrams with Three Circles. Presented at Euler Diagrams Workshop 2005, Paris (2005)
5. Pirooznia, M., Nagarajan, V., Deng, Y.: GeneVenn – a web application for comparing gene lists using Venn diagrams. Bioinformation, 1:420–422 (2007)
6. Wilkinson, L.: Venn and Euler Data Diagrams. Science (2), Citeseer (2010)
7. Healey, C., Booth, K., Enns, J.: High-speed visual estimation using preattentive processing. ACM TOCHI (1996)
8. Jock Mackinlay. Automating the Design of Graphical Presentations, ACM Transactions on Graphics, 5(2) (1986)

9. MacEachren, A.: How Maps Work: Representation, Visualization and Design (2004)
10. Spoerri, A.: InfoCrystal: A Visual Tool For Information Retrieval And Management. In: CIKM '93 Proceedings of the second international conference on Information and knowledge management (1993)
11. Clark, J.: TwitterVenn. www.neoformix.com/Projects/TwitterVenn/view.php
12. Brase, G.L.: Pictorial representations in statistical reasoning. Applied Cognitive Psychology, 23(3), 369-381 (2009)
13. Neurath, O.: International picture language. London: Kegan Paul (1936)
14. Rose, S.: Social Stratification in the United States: The American Profile Poster. The New Press (2007)
15. Jonker, D., Wright, W., Schroh, D., Proulx, P., Cort, B.: Information Triage with TRIST. 2005 Intelligence Analysis Conference (2005)
16. Ware, C.: Visual Thinking for Design. Ch. Structuring Two-Dimensional Space, pp. 43-65, Morgan Kaufmann (2008)
17. Byelas, H., Telea, A.: Texture-based Visualization of Metrics on Software Architectures. Software Visualization (2008)
18. Simonetto, P., Auber, D., Archambault, D.: Fully Automatic Visualisation of Overlapping Sets. Eurographcs/IEEE-VGTC Symposium on Visualization (2009)
19. Ware, C.: Information Visualization: Perception for Design. Ch. Glyphs And Multivariate Discrete Data, pp. 176-184 (2004)
20. Brath, R.: The Many Dimensions of Shape. IV'09 Opening Keynote, www.oculusinfo.com/expertise.html (2009)

Author Index