Available online at www.sciencedirect.com
SCIENCE DIRECT®

Electronic Notes in Theoretical Computer Science

ELSEVIER Electronic Notes in Theoretical Computer Science 134 (2005) 103–126
www.elsevier.com/locate/entcs

Measuring and Reducing Clutter in Euler Diagrams

Chris John

Visual Modelling Group
University of Brighton
Brighton BN2 4GJ, UK

Abstract

When modelling with three or more sets Euler diagrams can become crowded or cluttered and their ability to intuitively represent relationships between sets diminishes. Projections are a notation that bring syntactic efficiency to Euler diagrams and the flexibility to represent relationships between sets in a variety of semantically equivalent ways. This paper briefly outlines the first sound and complete system of Euler diagrams to incorporate the notation of projections. It defines a metric for measuring clutter in a diagram and outlines an algorithm that, given a diagram of the system, finds a semantically equivalent diagram(s) with the minimal measure of clutter.

Keywords: Projections, Clutter, congestion.

1 Introduction

Euler diagrams, or Euler circles, first appeared in 1761 [1]. Euler had the representation of classical syllogisms in mind, not sets, and his motivation appears to have been to offer a diagrammatic alternative to sentential logic: *'These circles...are extremely commodious for unfolding all the bloated mysteries of logic, which art finds it so difficult to explain'.*

Over a hundred years later John Venn [2] developed his diagrammatic structures for representing syllogisms and it is his name with which most people associate diagrams of this type today. The American philosopher and mathematician Charles Peirce developed several systems of diagrammatic reasoning

[1] Email: C.John@brighton.ac.uk

doi:10.1016/j.entcs.2005.01.035

[3], also exploiting the properties of exclusion and containment, while developing the notation of *x-sequences* to symbolize the existence of elements. In recent years advances in computing have accompanied a resurgence in the field of diagrammatic reasoning and a greater degree of formalism. Eric Hammer produced a simple, sound and complete system of Euler diagrams [4]. Sun-Joo Shin developed several systems based on Venn diagrams [5] and several systems of '*spider diagrams*' [6], [7], [8] based on Venn and Euler diagrams, have been developed at the University of Brighton.

The original notion of projections was proposed in [9], and developed in [10] and [11]. The notation grew from a desire to limit the congestion that besets Euler diagrams as more sets are considered. Informally, a projection is a set represented within a context (a sub-domain of the universe of discourse). Outside of its context the projection does not assert anything. A projection may be represented in one or many ways by curves called projected contours. First we outline the system of Euler diagrams that we will augment with projections and detail two of its weaknesses.

2 Euler diagrams

Euler diagrams use the topological properties of enclosure, exclusion and intersection to represent the set-theoretic concepts of subset, disjoint set and set intersection respectively, with shading representing the empty set. The following examples of the system informally illustrate the syntax, semantics and reasoning using transformation rules.

The LHS diagram of Figure 1 has three contours labelled A, B and C. By the enclosure of the contour labelled B inside the contour labelled A we can infer that the set represented by the contour labelled B is a subset of the set represented by the contour labelled A. Similarly, by the exclusion of the contours labelled A and C we can infer that the sets these contours represent are disjoint.

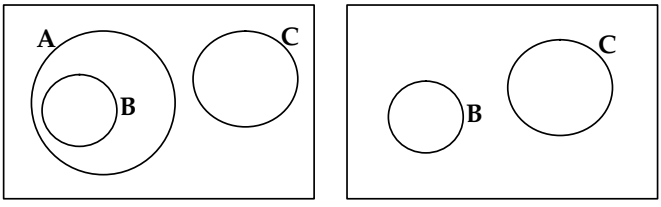

Fig. 1. Euler diagrams.

Just as we could infer from the premises $B \subseteq A$ and $A \cap C = \emptyset$, the conclusion $B \cap C = \emptyset$, so we can apply a simple reasoning rule *'Erase contour'* to the

contour labelled A to show the conclusion diagrammatically as in the RHS diagram of Figure 1. The diagrams are both clear and provide an intuitive representation of the information. However this is not always the case with Euler diagrams and we identify two key concepts that cause difficulties.

Inflexible representation

Let us consider the set of cars as our universal set and within this the set of American cars, the set of coupés and the set of blue cars. If we wish to use Euler diagrams to represent the statement 'All American coupés are blue', we could use a contour labelled A to represent the set of American cars, a contour labelled B to represent the set of blue cars, and a contour labelled C to represent the set of coupés. The diagrams of Figure 2 show the two ways of representing the statement. The LHS diagram uses a shaded region to show the set representing American coupés that are *not* blue, is empty. The RHS diagram similarly shows this set is empty by not having a region to represent it.

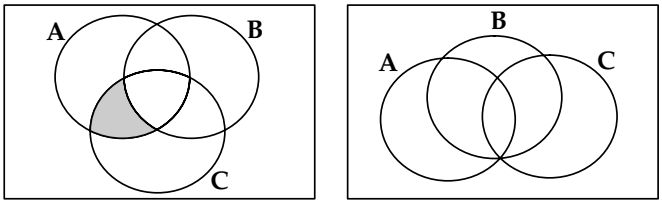

Fig. 2. Semantically equivalent Euler diagrams

The two statements 'All American coupés are blue' and 'No American coupé is not blue' have the same truth values but are nonetheless different sentences. Euler diagrams do not share the same flexibility of natural or more formal languages to express semantically equivalent statements in many syntactically distinct ways. This inflexibility becomes more acute as the number of sets under consideration increases.

A requirement to display all relationships

Another concern in Euler diagrams that rapidly becomes acute when considering more sets is that of 'congestion' or 'clutter'. An Euler diagram representing n sets represents all possible relationships (2^n) between these sets either by using an area of the diagram, or, by having no area of the diagram to represent the relationship we can infer that the set is empty.
The LHS diagram of Figure 3 contains 16 zones although the only information we can infer from the diagram is $A \cap B \cap C \cap D = \emptyset$. Similarly in the RHS

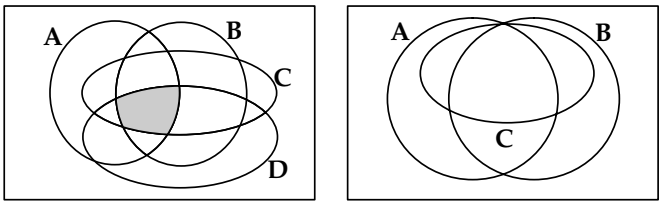

Fig. 3. Euler diagrams.

diagram of Figure 3 the diagram requires 7 zones to represent $C \cap \overline{A \cup B} = \emptyset$. Projections tackle both these weaknesses in Euler diagrams. They bring flexibility by providing many syntactically distinct but semantically equivalent representations. Additionally, they can greatly reduce syntactic congestion (which we shall call **clutter** in this paper) by allowing sets to be represented within a restricted domain. Before we show these qualities we give an abridged version of the syntax, semantics and reasoning rules of 'Euler diagrams with projections', or $\mathcal{EDP}$ for short.

3 Euler diagrams with projections

We give a brief description of the main features of $\mathcal{EDP}$.

Concrete syntax

A concrete Euler diagram with projections consists of **given contours** which have unique labels and **projected contours**, drawn here with dashed iconography, which may not have unique labels. All contours are properly contained within a **boundary rectangle**. The intersection, containment or exclusion of the contours partitions the area within the boundary rectangle into unique **zones** which may be shaded. A non-empty set of zones is a **region**.

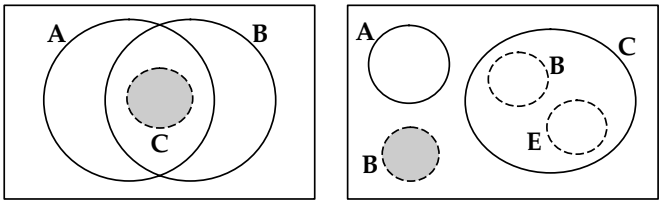

Fig. 4. Euler diagrams with projections.

In the LHS diagram of Figure 4 the given contours are labelled A and B while the projected contour is labelled C. The diagram has five zones, one example is the region of the diagram inside the contour labelled B and outside the contour labelled A. This diagram has only one shaded zone, the zone inside

the projected contour labelled C. The RHS diagram of Figure 4 has two projected contours labelled B.

Abstract syntax

Diagrammatic systems usually have a two-tiered syntax [12], a diagrammatic representation in some media (concrete syntax) and an abstraction, which forgets the geometric information and retains only the essential relationships (abstract syntax). The notation of projections suits a three-tiered syntax, splitting the abstract syntax into a fine-grained and coarse-grained levels. The fine-grained abstract syntax has the detail for applying reasoning rules while the coarse-grained abstract syntax gathers together many fine-grained types that share the same interpretation.

Fine-grained abstract syntax

The fine-grained abstract syntax records how a diagram uses projected contours to represent a projection. The fine-grained abstract syntax includes a set of **underlying zones** (an underlying zone is a zone defined in terms of given contour labels only). A projected contour in the fine-grained abstract syntax is a pair, the first element being the label and the second being the set of underlying zones (an underlying region) which the projected contour intersects. This region is called the **context** of the projected contours. Any two projected contours sharing a label will have disjoint contexts.
In the LHS diagram of Figure 5 there are two projected contours used to represent the projection of the set represented by the label E. The RHS diagram of Figure 5 is identical to the LHS diagram except that it uses only one projected contour to represent the projection of the set represented by the label E.

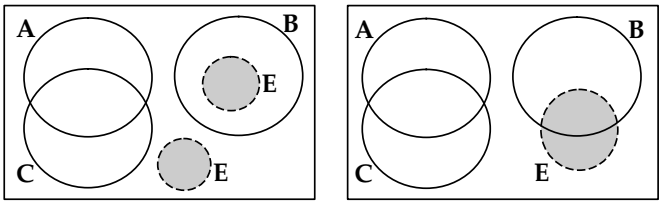

Fig. 5. Diagrams with different fine-grained abstractions.

Coarse-grained abstract syntax

The two diagrams of Figure 5 have a different fine-grained abstract syntax but the same coarse-grained abstract syntax as this level forgets the way that a

projection is represented by projected contours. The notion of contour is lost but the ideas of underlying zone and therefore context are carried on.

Semantics

The interpretation of projected contours used in this system first appeared in [10] although alternative semantics have been proposed [11] and others could no doubt be envisaged. The interpretation is called **'underlying region'** semantics. The underlying zones of the diagram form underlying regions and the underlying region that a projected contour label intersects with is called the **context** of the projected contour label. Therefore, any projected contour label is interpreted as:
'The intersection of the set assigned to the label of the projected contour with the set assigned to the context of the projected contour label.'

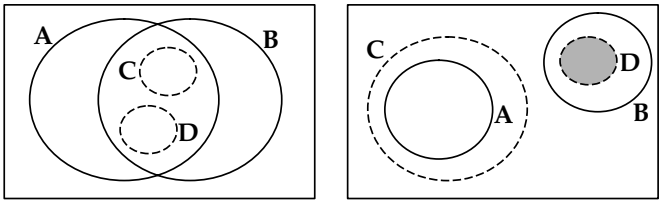

Fig. 6. Illustrating the semantics of projected contour labels.

The LHS diagram of Figure 6 has four underlying zones: one outside both the contours labelled A and B, one inside the contour labelled A but outside the contour labelled B, one inside the contour labelled B but outside the contour labelled A and finally, one inside both the contour labelled A and the contour labelled B.
This last underlying zone is the context of the projected contours labelled C and D and their exclusion within this underlying zone implies that, within the intersection of the sets represented by the labels A and B, the intersection of the sets represented by the labels C and D is empty, or

$$A \cap B \cap C \cap D = \emptyset$$

In the RHS diagram of Figure 6 the context of the projected contour label C is the underlying region outside the contour labelled B, while the context of the projected contour label D is the underlying region inside the contour labelled B. As these two projected contours have disjoint contexts the diagram does not provide any information regarding the relationship between the sets represented by the labels C and D.

Reasoning rules

Here we informally introduce four reversible rules of particular relevance to the transformation of projected contours.

- **Introducing a missing zone.**
  This rule allows for the introduction of a shaded zone to represent a zone that was missing due to the containment or exclusion of contours. In the LHS diagram of Figure 7 the projected contours labelled B and C are disjoint within the underlying zone inside the given contour labelled A. In the RHS diagram a shaded zone has been introduced to represent their empty intersection.

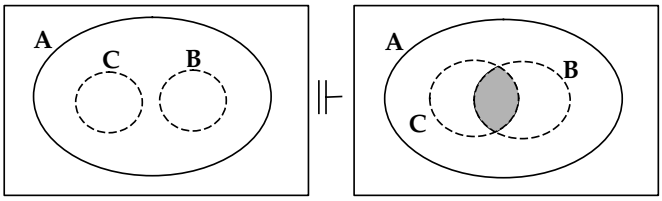

Fig. 7. A valid application of 'Introducing a missing zone'.

Note that not all containment or exclusion in a diagram indicates that zones are missing. In the LHS diagram of Figure 8 the contours labelled B and C are disjoint but we may not infer that their intersection is empty. As the contour labelled C is projected, it is only interpreted within its context, which is the underlying zone inside the contour labelled A and outside the contour labelled B.

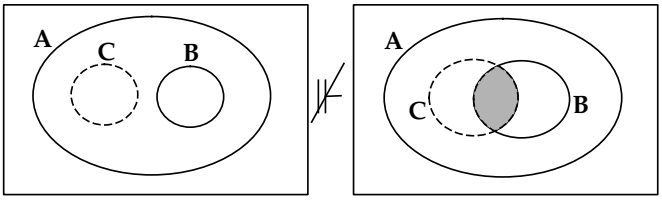

Fig. 8. An invalid application of 'Introducing a missing zone'.

- **Introducing a projected contour.** This rule allows for the introduction of a projected contour into any underlying zone that does not already contain one with the same label.

  In the LHS diagram of Figure 9 there are two underlying zones. The rule would not allow the introduction of a projected contour labelled B (or C) into the underlying zone *within* the given contour labelled A, as there already exists one. It does allow the introduction of a projected contour

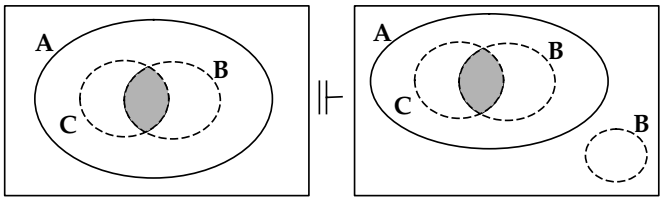

Fig. 9. Introducing a projected contour.

labelled B into the underlying zone *outside* the given contour labelled A, as has been done in the RHS diagram.

- **Merge two projected contours.** This rule allows for two projected contours with the same label to be merged into one projected contour.

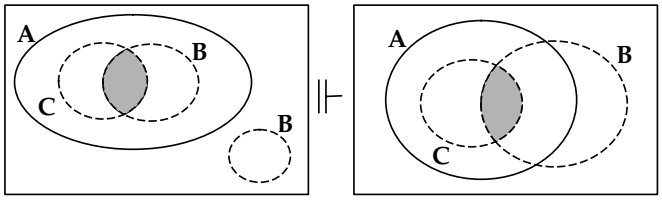

Fig. 10. Merging two projected contours.

In the LHS diagram of Figure 10 there are two projected contours labelled B. In the RHS diagram they have been replaced with a single projected contour labelled B.

- **Replacing a projected contour with a given contour.** This rule allows for changing of a projected contour into a given contour under the following conditions:

(i) The context of the projected contour is the set of underlying zones.
(ii) In every underlying zone that the projected contour does not contain, it splits every zone.

The LHS diagram of Figure 11 has a projected contour labelled B that satisfies the conditions needed to allow it to be replaced with a given contour, as has been done in the RHS diagram. Note that the projected contour labelled C does not satisfy the first condition.

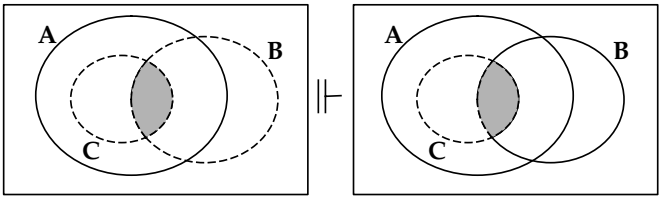

Fig. 11. Replacing a projected contour with a given contour.

The reader may have noticed that applying these four rules has changed the projected contour labelled B in the LHS diagram of Figure 7 into a given contour labelled B in the RHS diagram of Figure 11. In general these four rules are all that is needed to change *any* projection into a given contour and it is this process that forms the basis of the soundness and completeness theorems of the system (omitted here for space considerations).

Additionally, the rule 'Merge two projected contours' can be used to define the coarse-grained abstract syntax. The rule defines an equivalence relation on the set of fine-grained abstract diagrams. This produces equivalence classes within which every diagram has the same coarse-grained abstraction.

Having outlined the main components of the system we return to justify the claims made earlier, that projections can add syntactic flexibility to and reduce congestion in Euler diagrams while preserving semantics.

Flexible representation

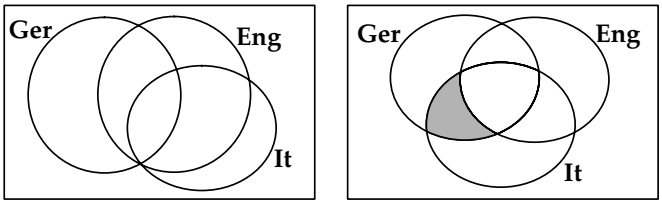

Fig. 12. Two Euler diagrams without projections.

Consider as the universal set the attendees to a particular conference and the sets of German, Italian and English speakers among the attendees. The statement 'All those who spoke German and Italian also spoke English' would be represented by a diagram similar to those in Figure 12, were we not to use projections.

If we were to consider using projections we may have many more semantically equivalent but syntactically distinct representations to choose from. Four of these are shown in Figure 13. Each diagram is saying the same thing in a different way just as the natural language statement 'Nobody spoke German and Italian and not English' is semantically equivalent to our original statement 'All those who spoke German and Italian also spoke English'.

Reducing clutter

Whether a diagram appears cluttered or congested is often a subjective matter. However, we argue that in Figure 14, the LHS diagram is considerably more congested than the RHS diagram.

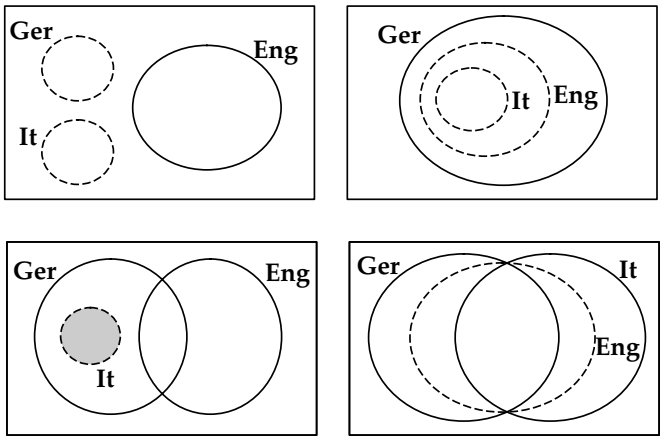

Fig. 13. Four semantically equivalent Euler diagrams with projections.

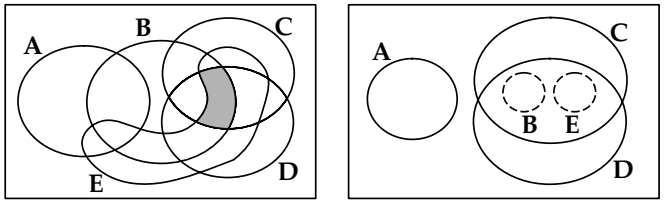

Fig. 14. Two semantically equivalent Euler diagrams with projections.

These diagrams are semantically equivalent and one can be transformed into the other using the four equivalence rules defined earlier. The RHS diagram exploits the topological properties of enclosure and exclusion to produce a less congested diagram, removing a lot of the intersections that makes the LHS diagram appear so cluttered. We would like to be able to automate the process by which we obtained the RHS diagram from the LHS diagram in Figure 14 and ideally we would like any algorithm to calculate a 'minimally cluttered' diagram. To quantify clutter in the diagrams of $\mathcal{EDP}$ we must first define what we mean by clutter and derive a metric to measure the level of clutter in each diagram.

4 Clutter in Euler diagrams with projections

A standard dictionary definition of the word 'clutter' includes the terms:
'an untidy collection of objects', 'filling space in a disorderly way', 'to crowd together in disorder', 'a confused multitude of things'.
Often a cluttered room is used as metaphor. There seems to be a subtle distinction within the definition of this word whether as verb or noun. The first two terms indicate that clutter is the result of *how* a collection of things

are arranged. One could imagine this type of clutter describing a teenager's room, which is often only cluttered in the sense that the items are strewn randomly about the floor as if the place had recently been burgled. The last two terms point to a more inherent cause to the clutter that may not be solved by simply arranging items in a more orderly manner, as in an overcrowded warehouse for example. In this section we argue that this distinction extends to the syntax of Euler diagrams and helps us to distinguish a type of clutter that we can clearly define and measure.

Concrete clutter

Informally we define **concrete clutter** as any clutter that does not abstract to the abstract syntax. Consider the LHS diagram of Figure 15.

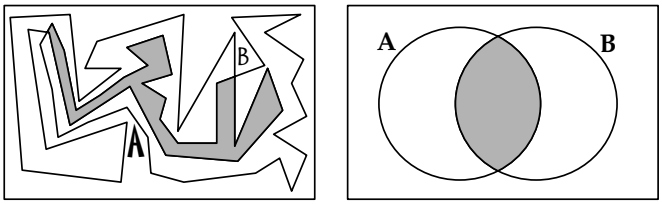

Fig. 15. Illustrating concrete clutter.

Apart from resembling a rather bad piece of 'cubism' it is nonetheless a well-formed Euler diagram in that it consists of simple closed curves that are uniquely labelled. However, as the contours are very irregular in shape, the labels have different fonts and sizing and are ambiguous in their placement as to which contour they are labelling, interpretation of this diagram is made unnecessarily difficult.

Conversely the RHS diagram has regular circular contours, clear labelling and, in a platonic sense, is probably quite close to the 'ideal' concrete representation of its type. So while both these diagrams have the same abstract syntax they differ greatly at the concrete level.

Abstract clutter

Now consider the LHS diagram of Figure 16. The contours are unlabelled not just for the purpose of illustration but also because it is very difficult to clearly label this diagram as the nature of the relationships force the contours to become smaller and closer together. This is not the result of bad drawing practices (we argue) but is inherent in the relationships between the contours and is an example of **abstract clutter**.

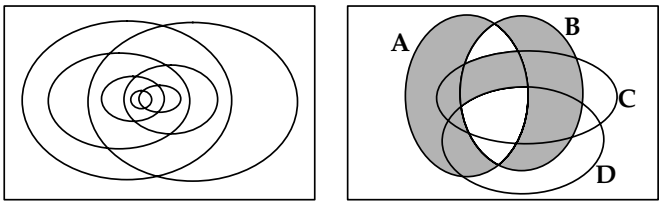

Fig. 16. Illustrating abstract clutter.

The RHS diagram of Figure 16 is another example of abstract clutter in that there is a number of shaded zones in the diagram, and this property will be present however we choose to draw it. This, coupled with the fact that the contours are densely overlaid, results in a cluttered diagram. These properties will be present in all concrete representations and it is precisely these properties that constitute abstract clutter. Having argued the case for a distinction in types of clutter in Euler diagrams we now look more closely at the factors that contribute to abstract clutter (for the sake of brevity shortened to 'clutter' hereon) in more detail.

Factors contributing to clutter

At first sight there appear to be many possibilities for factors. One of the most obvious is the number of zones and although this is not subtle enough to capture all the variations in clutter it remains a very good single measure.

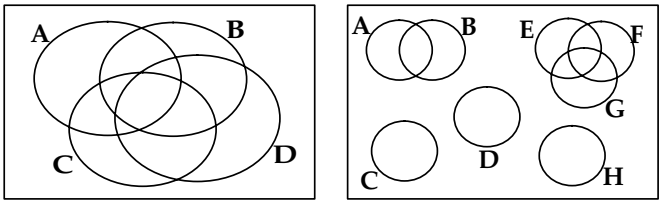

Fig. 17. Factors contributing to clutter.

The number of components is clearly a factor as the diagrams in Figure 17 attest to. Both diagrams have the same number of zones but we argue that the greater number of components in the RHS diagram renders it less cluttered in comparison to the LHS diagram, even though it has twice as many contours and labels.

Each of the diagrams in Figure 18 contain a single component (a connected subgraph) composed of five contours creating nine zones (the minimum for a connected component). These four diagrams can be seen as patterns or types that can be extended to any number of contours greater than two.

We argue that the diagrams on the top row are less cluttered than the dia-

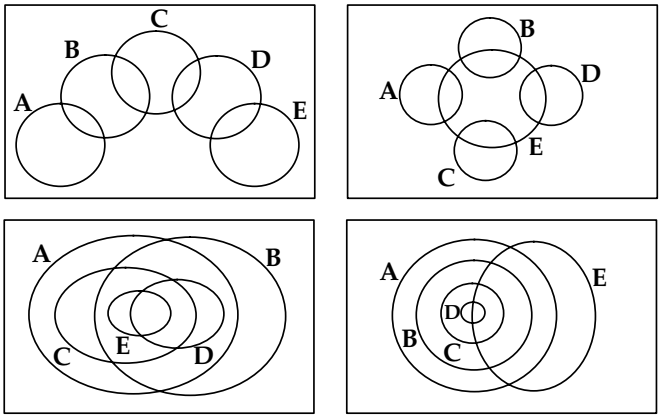

Fig. 18. All components have 5 contours and 9 zones.

grams on the bottom row. In particular we argue that the difference in clutter becomes more pronounced as the number of contours increases. The explanation for this appears to be the properties of **pair-wise disjointness** and **degree of containment** between the contours in the diagram. In a concrete sense two contours are pair-wise disjoint if their interiors have no common point and a contour is said to be contained by degree n if it is wholly contained within n contours. The diagrams in the top row of Figure 18 have the maximum number of pair-wise disjoint relations for a connected component, $(n-1)(n-2)/2$ for $n > 2$ contours, and the minimum degrees of containment (zero). Conversely, the diagrams in the bottom row of Figure 18 have the minimum number of pair-wise disjoint relations for a connected component (zero) and the maximum degrees of containment, $(n-1)(n-2)/2$ for $n > 2$ contours. As all other factors in the diagrams are identical we conclude that containment increases clutter more than disjointness.

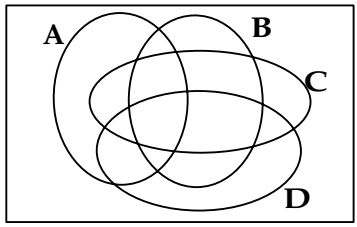

Fig. 19. Maximum intersection between 4 contours.

The only other factor to consider between contours is **intersection**, of which all the diagrams of Figure 18 have the minimum number. It is not quite so easy to compare the relation of intersection with either pairwise-disjoint or degree of containment because intersection always results in the creation of

extra zones between two contours, while the other two relations do not. The diagram in Figure 19 is a representation of a Venn diagram on four contours. It has no pair-wise disjoint relations or containment but the maximum number of intersections between the contours of the diagram. Intersection between contours increases the number of zones in a diagram, causing the contours to overlay and contribute clutter to a component. It is evident that, when considering the relation between two contours, intersection adds more clutter than one contour being contained within another while both add more clutter than two contours being disjoint.

We have argued that the three relations between contours in a diagram contribute different quantities of clutter to the components of a diagram. However, we are still left with a lengthy task to extract this information from the diagram. In addition to the task of identifying the components in a diagram and the relations *within* them, we then need to calculate the variation in clutter *between* components, by extending the argument (from contours to components) that contained components will contribute more to the clutter of a diagram than those that are disjoint.

Contour scoring

Although the preceding section appears to identify several factors that are time-consuming to measure, there exists a mechanism that is both intuitive at the concrete level and simple at the abstract level and measures the same factors identified as the causes of clutter in components of a diagram. It also measures the clutter between components. The method is called contour scoring and is deceptively simple. We illustrate the method at the concrete level using the diagrams of Figure 20.

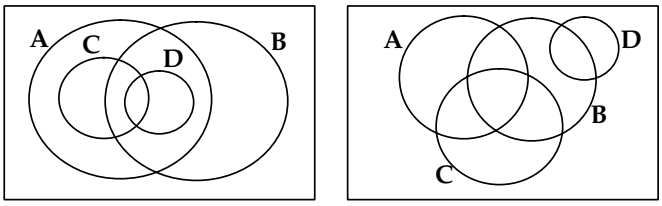

Fig. 20. Illustrating contour scoring

To contour score these diagrams we simply count the number of zones within each contour and sum. For the LHS diagram there are six zones inside the contour labelled A, five inside the contour labelled B and three and two zones inside the contours labelled C and D respectively. Therefore the contour score for the LHS diagram is 16. A similar calculation yields a contour score for the RHS diagram of 15. Therefore, by the measure of contour scoring these two

diagrams have similar levels of clutter. Note that the RHS diagram has more zones than the LHS diagram but a lower contour score.

The contour scoring mirrors the factors identified earlier and their respective contribution to clutter. If two contours are disjoint there is no overlap and no zones within these contours will contribute to the contour score of the other. If a contour has degree of containment n then all the zones within it will be counted n times in the contour score. Furthermore if two contours intersect they produce an extra zone within each contour. The relations between components is also reflected by this method of scoring.

Interestingly, the top two diagrams in Figure 18 score 13 by contour scoring while the bottom two diagrams both score 25. Generalized to n contours, the top two patterns will have a contour score of $3n - 2$ while the bottom two will score n^2. This reflects that the clutter in these two pairs of diagram (that have the same number of zones and contours) is different and will diverge as the number of contours increases. A Venn diagram on n contours will have a contour score of $n2^{n-1}$, resulting in the diagram in Figure 19 having a contour score of 32.

We therefore propose a weight-function, $\mathcal{CS}$, for measuring clutter in the diagrams of $\mathcal{EDP}$. It is composed of two parts, the contour score and the number of shaded zones in the diagram:

$$\mathcal{CS}(d) = \sum_{(x,y)\in Z(d)} |x| \;+\; \alpha|Z^*(d)|$$

Note that the constant, α, will be taken as unity until a more accurate estimate can be found from empirical study.

Zone scoring and the dual graph

An alternative idea for capturing the subtle factors that contribute to clutter involves the use of the dual graph and an equivalent method called zone scoring. **Zone scoring** involves scoring each zone according to the number of contours that contribute to its perimeter. The diagrams in Figure 21 illustrate the methods with the RHS diagram showing each zone scored for the zones of the LHS diagram.

The graph in the LHS diagram of Figure 21 is the dual graph of the diagram and, by summing the vertex degrees we arrive at the same score as the zone scoring method. Both (equivalent) methods are trying to measure the complexity of a zone in the sense that, in order to interpret a zone in a diagram we have to first ascertain which contours it is contained and excluded by. However it does not count the edges of a zone (one zone in Figure 21 has a score of

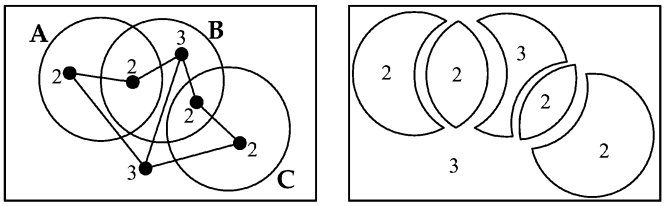

Fig. 21. Illustrating zone scoring

3 but has 4 edges) and, in not counting the contours that do not contribute to its perimeter, assumes that containment and disjointness both 'come for free'.

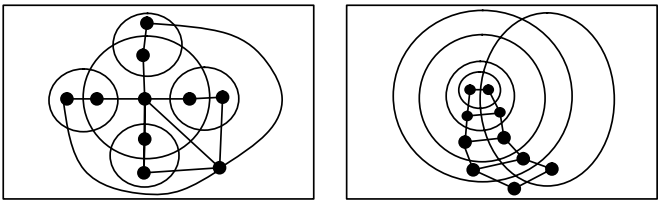

Fig. 22. Two diagrams and their dual graphs

More importantly, this method does not distinguish between the diagrams in Figure 22. Either by zone scoring or counting the vertex degrees of the dual graph, both diagrams score the same. Therefore these methods cannot account for the variation in clutter due to pair-wise disjointness or degree of containment between contours. Zone scoring is less sensitive to the relations between the contours in a component and requires more work to calculate.

5 Automated Reduction of clutter in Euler diagrams with projections

The ability of projections to reduce clutter in Euler diagrams has been illustrated in earlier papers [10], [11] and throughout this paper. With the development of a metric to measure clutter defined on the diagrams of $\mathcal{EDP}$ we now seek an algorithm that, given a diagram, will return a diagram(s) that is semantically equivalent and has the lowest $\mathcal{CS}$. Ideally, as a major strength of projections is their flexibility of representation, it would be a useful by-product of any algorithm to produce a selection of diagrams with different syntactic representations.

'Minimize contour label'

We outline an algorithm that attempts to do these two tasks in the next section. The main component of the algorithm is the derived rule **'Minimize**

contour label'. We illustrate this rule with the diagrams of Figure 23.

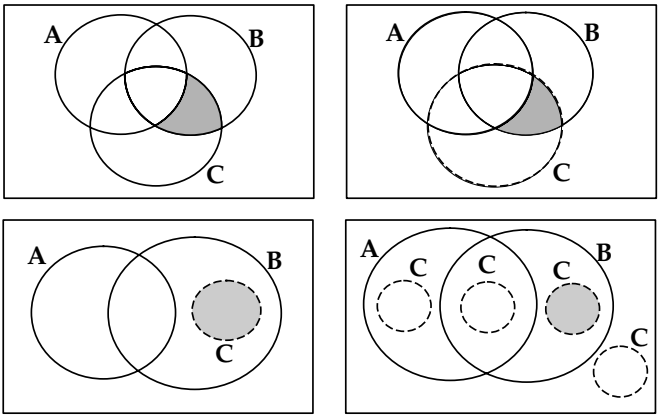

Fig. 23. Illustrating the derived rule 'Minimize contour label'

Reading the diagrams clockwise from top LHS, our initial diagram has three contour labels and we wish to apply the rule 'Minimize contour label' to the contour label C. The first part of the rule replaces the given contour labelled C with a single projected contour labelled C. The underlying zones of the diagram have now changed and by the repeated use of the 'split projected contour in two' rule (the reverse of 'merge two projected contours') we have a projected contour in every underlying zone. For every projected contour labelled C we test whether we can apply the reverse of 'Introducing a projected contour'. Projected contours that the reverse rule can be applied to provide no semantic contribution and can therefore be erased, while those that the rule cannot be applied to *do* carry semantic information. In this example we can apply the reverse rule to three of the four projected contours resulting in the final diagram of Figure 23. This rule is always applied to a given contour label and results in the label being projected into the smallest underlying region. Finally we apply the reverse of 'Introduce missing zone' to the diagram wherever possible to remove any missing zones that may have been created. Applying the rule to the contour label C has reduced the $\mathcal{CS}$ from 13 to 7, but this is not the minimally cluttered diagram that we are looking for (although it may be useful as an alternative, semantically equivalent representation with a lower $\mathcal{CS}$ than the initial diagram). If we apply the rule 'Minimize contour label' to the contour label B we arrive at the diagram in Figure 24.

We can be sure this is a minimally cluttered diagram from all the contours being disjoint, resulting in $\mathcal{CS} = 3$ (as every contour contributes at least 1 to the $\mathcal{CS}$, a lower score is not possible). We now give an informal description of a proposed algorithm involving the rule 'Minimize contour label' (denoted

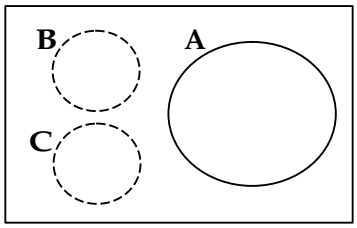

Fig. 24. A minimally cluttered diagram.

$Min(L)$ in the diagrams).

CSA-An algorithm to minimize clutter

Let d be a (unitary) diagram of $\mathcal{EDP}$ (a unitary diagram is just a single box). We call d the **initial diagram** and calculate the $\mathcal{CS}$ for d (as we do for all diagrams produced by the algorithm). The first diagram to be calculated is called the **header diagram**, denoted d_g. The initial diagram is transformed into d_g by performing two derived rules in sequence:

(i) **Replace all projections.** This rule is a combination of the four rules highlighted previously and results in all projections in d being replaced with given contours.

(ii) **Remove all missing zones.** This rule is merely the repeated use of the reverse of the rule 'Introduce missing zone'.

The LHS diagram of Figure 25 illustrates the initial diagram while the RHS diagram shows the header diagram. For illustration $\mathcal{CS}(d) = 12$ while $\mathcal{CS}(d_g) = 10$.

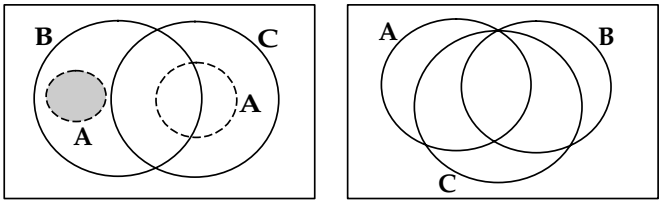

Fig. 25. Initial diagram, d, and header diagram, d_g.

The header diagram is a sort of canonical form in that we know all the contours are given contours and the shaded zones have been reduced to a minimum. We use this diagram as a base from which to produce the first set of diagrams. Each diagram in this set involves applying the rule 'Minimize contour label' to exactly one contour label of d_g. The diagrams in Figure 26 illustrate the set of diagrams that have one minimized contour label.

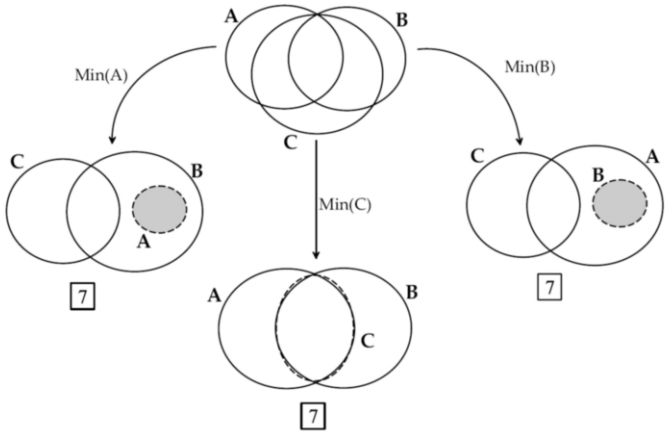

Fig. 26. The set of diagrams with one minimized projection (clutter scores in boxes below).

The header diagram has three contour labels and so the set will contain three diagrams. Below the diagrams is their $\mathcal{CS}$. Note that applying 'Minimize contour label' to the contour label C gives a diagram with concurrent contours (these are only drawn slightly overlapping here to highlight the issue) which shows that minimizing projections may result in a diagram that has no well-formed concrete representation.

The next stage is to apply the rule 'Minimize contour label' to another given contour label in each of the diagrams of the set that have one minimized projection to give the set of diagrams that have two projected labels. The diagrams of Figure 27 illustrate the process.

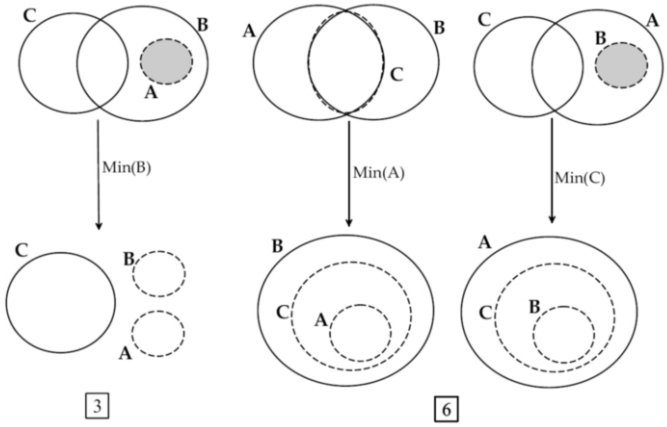

Fig. 27. The set of diagrams with two minimized projections (clutter scores below).

This process is continued step by step, with the rule 'Minimize contour label' applied to an additional given contour label each time to give a new set of

diagrams, until all contours are projected. Let us call this diagram d_p. The diagram d_p will have the same structure as the header diagram d_g except that all the contours will be projected contours. Consequently, as the metric $\mathcal{CS}$ does not distinguish between the type of contour in a diagram, $\mathcal{CS}(d_g) = \mathcal{CS}(d_p)$, so d_p need not be calculated. We include this type of diagram in Figure 28 for illustration only. The diagrams of Figure 28 show the algorithm (the set of diagrams and clutter scores) with respect to the initial diagram of Figure 25.

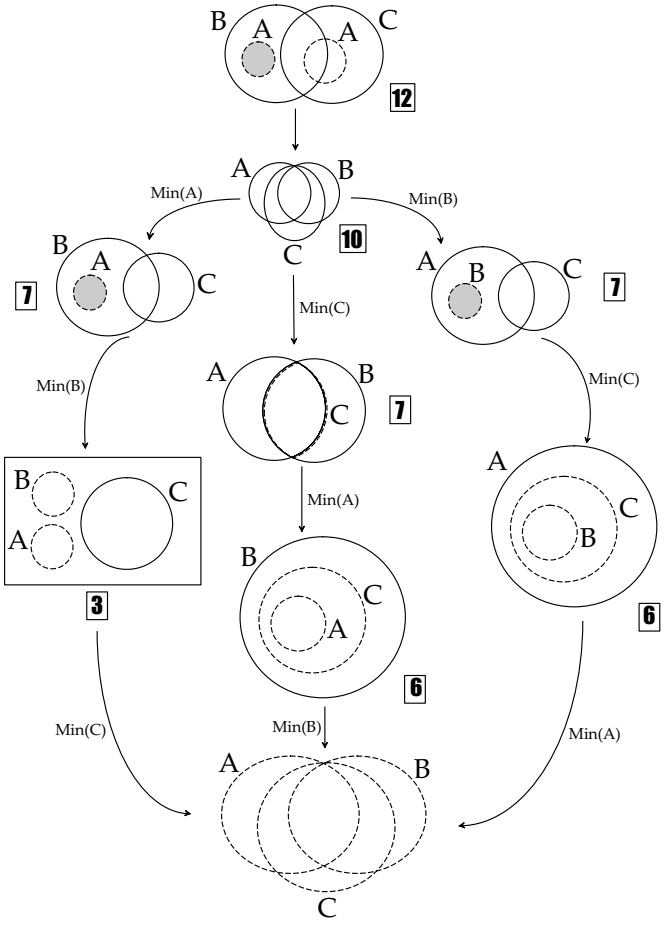

Fig. 28. The entire set of diagrams of the search space (with clutter scores) with the minimally cluttered diagram boxed.

Search space for CSA

We define the **search space** of $\mathcal{CSA}$ as:
'The set of diagrams that the algorithm must compute (and score for clutter) in order to ensure that a (semantically equivalent) diagram with the minimal $\mathcal{CS}$ has been found'.
We give the following conjecture as to properties of the rule 'Minimize contour label' that will limit the size of, and classify, the search space for a diagram of $\mathcal{EDP}$.

Conjecture 5.1 ***'Minimize contour label' is commutative.***

This result would reduce the size of the search space to 2^n diagrams for an initial diagram with n contour labels.
We now describe an automorphism on the contour labels of the diagram that, if present in the header diagram, reduces the search space further.

An automorphism between contour labels

Consider the diagrams of Figure 29. If we swapped the contour labels A and B in the LHS diagram the abstract syntax would be exactly the same. This is not true of the labels A and B in the RHS diagram, as swapping these labels *would* alter the abstract syntax.

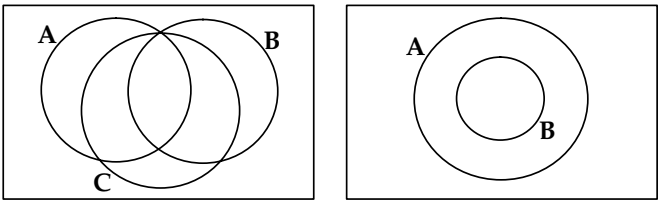

Fig. 29. Illustrating a label automorphism that can reduce the search space of $\mathcal{CSA}$.

Informally, if two labels can be swapped without changing the abstract syntax then the labels are automorphic and this enables the algorithm to be able to use one label to predict the contour score of the other. In Figure 28 the labels A and B in the header diagram are automorphic. We would expect the the algorithm to calculate eight diagrams (Figure 28 shows all these diagrams).
The automorphism between the contour labels A and B allows the algorithm to deduce the diagram (and $\mathcal{CS}$) that results from $Min(A)$ from $Min(B)$. Similarly the diagram and $\mathcal{CS}$ that is the result of $Min(A)Min(C)$ gives the diagram resulting from $Min(B)Min(C)$ and so on. In this way, along with the fact that $\mathcal{CS}(d_g) = \mathcal{CS}(d_p)$, the algorithm need only calculate five diagrams (and score the initial diagram d) to conclude. The reduced search space is shown in Figure 30.

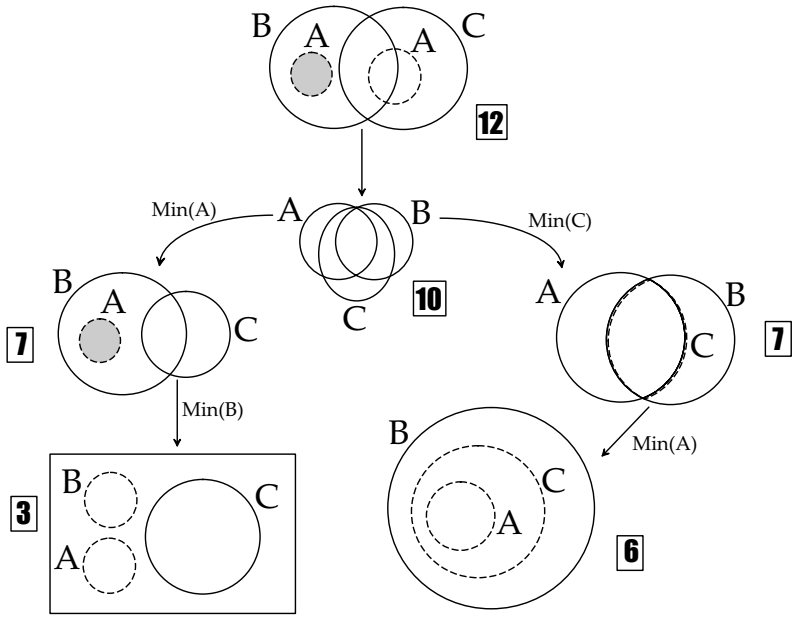

Fig. 30. The reduced set of diagrams of the search space.

The previous example is useful for illustrating $\mathcal{CSA}$ but it should be noted that not all diagrams reduce in such a straightforward manner when the algorithm is applied. Let the diagram in Figure 31 be the initial diagram d.

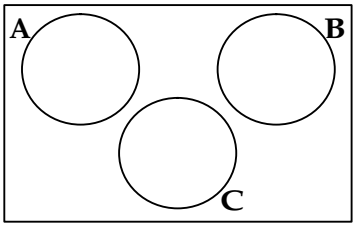

Fig. 31. An initial diagram, d.

The search space of d is shown in Figure 32. As the contours in d are all given contours and there is no shading $d = d_g$ and the diagram is scored for clutter, $\mathcal{CS}(d) = 3$. As all pairs of contour labels are automorphic, the algorithm need only calculate one diagram for each set. The diagram representing the set of diagrams with one contour label projected increases the $\mathcal{CS}$ from 3 to 8. Projecting two labels only serves in increasing the $\mathcal{CS}$ to 9 and the only other diagram to consider is d_p which we know has the same structure and clutter score as the header diagram. Therefore the algorithm found that the minimally cluttered diagram with respect to d was itself, illustrating that the rule

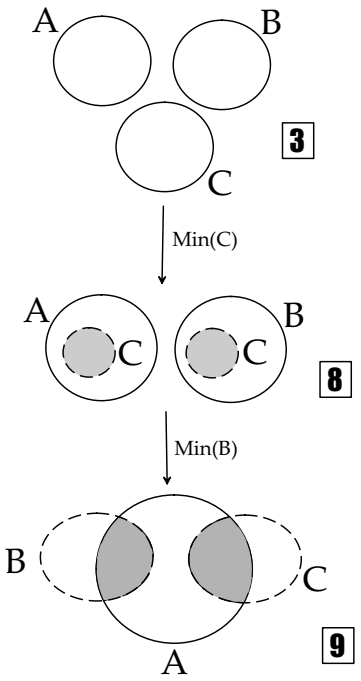

Fig. 32. The search space of d.

'Minimize contour label' may not always reduce the clutter score. However, it is conjectured that $\mathcal{CSA}$ will always contain the minimally cluttered diagram within its search space.

The examples of $\mathcal{CSA}$ shown in this paper both yielded well-formed diagrams but in general there is no guarantee this will be the case. A method of backtracking, introducing missing zones until a well-formed diagram is achieved, should be built into the algorithm for it to be of consistent practical use.

6 Conclusions and further work

In this paper we have tried to give an informal overview of several areas of current work concerning the notation of projections in Euler diagrams. The area covered has meant a lack of formal detail but stands as a forerunner for several future papers dealing in detail with each of the main areas covered here.

That projections can reduce syntactic congestion in many Euler diagrams is clear to see. This paper has shown their flexibility to represent logical statements in a variety of semantically equivalent ways. It should also be noted that projected contours are not only a notation for displaying semantic

equivalence but can be used diagrammatically to reason with in the same capacity as given contours. This will be expanded on in future work. However, it remains to be seen whether the benefits of projected contours outweigh the additional complexity of having two types of contour. How intuitive the semantics are to a user is a question for user studies and empirical research. Further work includes the empirical testing of $\mathcal{CS}$, the metric for clutter, and formal proofs of the properties of the clutter reduction algorithm, $\mathcal{CSA}$. Euler diagrams are a basis for the more expressive constraint diagrams [9] and issues concerning the addition of projections to this system will also be investigated.

Acknowledgment

Chris John thanks the UK EPSRC for support under grant number **02800109** and John Howse, John Taylor, Jean Flower and Gem Stapleton for comments on earlier drafts on this paper.

References

[1] Euler, L., *Lettres a Une Princesse d'Allemagne*, vol 2, 1761, Letters No. 102-108. Venn:1880,

[2] Venn, J., *On the diagrammatic and mechanical representation of propositions and reasonings*, Phil.Mag, 1880.

[3] Peirce, *Collected Papers*, Vol. 4., Harvard Univ. Press, 1933.

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

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

[6] Howse, J., Molina F. and Taylor J., *A sound and complete diagrammatic reasoning system*, Proceedings of IEEE Symposium on Visual Languages (VL2000), IEEE Computer Society Press,127–136, 2000.

[7] Howse J., Molina F., Taylor J., Kent S. and Gil J., *Spider Diagrams: A Diagrammatic Reasoning System*, J. of Visual Languages and Computing, 2001.

[8] Stapleton, G., *Reasoning with constraint diagrams*, PhD thesis, University of Brighton, 2004.

[9] Gil J. and Howse J. and Kent S., *Constraint Diagrams: A step beyond UML*, Proceedings of TOOLS USA 1999 Santa Barbara, California, USA, IEEE Computer Science Press, 453-463,1999.

[10] Gil J. and Howse J. and Kent S. and Taylor J., *Projections in Venn-Euler Diagrams*, Proceedings of IEEE Symposium on Visual Languages (VL2000), IEEE Computer Science Press, 119-126, 2000.

[11] Gil J. and Howse J. and Tulchinsky E., *Positive semantics of projections*, Journal of Visual Languages and Computing, volume 13(2), 197-227, April, 2001.

[12] Howse J., Molina F., Shin S-J. and Taylor J., *On Diagram Tokens and Types*, Proceedings of Diagrams 2002, Calloway Gdns, Georgia, Springer-Verlag, 146-160, April, 2002.