Electronic Communications of the EASST
Volume 7 (2007)

Proceedings of the Workshop on the Layout of (Software) Engineering Diagrams (LED 2007)

Properties of Euler Diagrams

Gem Stapleton, Peter Rodgers, John Howse and John Taylor

15 pages

Guest Editors: Andrew Fish, Alexander Knapp, Harald Störrle
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Properties of Euler Diagrams

Gem Stapleton[1], Peter Rodgers[2], John Howse[3] and John Taylor[4]

[1] g.e.stapleton@brighton.ac.uk,[3] john.howse@brighton.ac.uk
[4] john.taylor@brighton.ac.uk
www.cmis.brighton.ac.uk/reseach/vmg
University of Brighton, UK

[2] p.j.rodgers@kent.ac.uk
University of Kent, UK

Abstract: Euler diagrams have numerous application areas, with a large variety of languages based on them. In relation to software engineering, such areas encompass modelling and specification including from a formal perspective. In all of these application areas, it is desirable to provide tools to layout Euler diagrams, ideally in a nice way. Various notions of 'niceness' can be correlated with certain properties that an Euler diagram may or may not possess. Indeed, the relevant layout algorithms developed to date produce Euler diagrams that have certain sets of properties, sometimes called well-formedness conditions. However, there is not a commonly agreed definition of an Euler diagram and the properties imposed on them are rarely stated precisely. In this paper, we provide a very general definition of an Euler diagram, which can be constrained in varying ways in order to match the variety of definitions that exist in the literature. Indeed, the constraints imposed correspond to properties that the diagrams may possess. A contribution of this paper is to provide formal definitions of these properties and we discuss when these properties may be desirable. Our definition of an Euler diagram and the formalization of these properties provides a general language for the Euler diagram community to utilize. A consequence of using a common language will be better integration of, and more accessible, research results.

Keywords: Information visualization, software specification, Venn diagrams

1 Introduction

In software engineering, a large variety of diagrammatic languages are used to model and reason about software. Of these languages, many are based on closed curves including some components of the Unified Modeling Language (UML) such as class diagrams and state charts [DC05]. Indeed, for the formal specification of software, Kent introduced constraint diagrams [Ken97] which are based on the well-known Euler diagram language. Constraint diagrams can express system invariants and operation pre-conditions and post-conditions. An example can be seen in figure 1 which shows an invariant on a video rental system, expressing that titles are partitioned into two disjoint subsets. The asterisks are universal quantifiers and the arrows, together with their sources and targets, make statements about properties of binary relations. In this particular

example the arrows are both labelled *NC*, for 'number of copies'. The diagram thus asserts that all elements in the set *ExColl* (ex-collection) have no associated copies; all other titles are 'in-collection' and are associated with some number of copies distinct from zero. The four closed curves in this diagram constitute and Euler diagram.

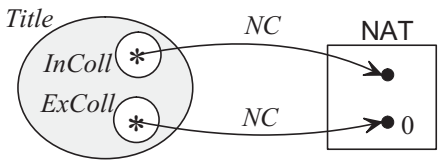


Figure 1: A constraint diagram.

Constraint diagrams fulfil a similar role to that of the Object Constraint Language, the only non-visual part of the UML; see [HS05, KC99] for examples of software modelling using constraint diagrams. A well studied fragment of the constraint diagram language, called spider diagrams [HST05], which are also based on Euler diagrams has been used in a variety of application areas; see, for example, [Cla05] where they are used in a safety critical environment. The relationship between spider diagrams, regular languages and finite state machines has also been investigated [DS07]. Euler based diagrams have numerous other application areas; for example [DES03, HES+05, KMGB05, Lov02, SA05, TVV05].

Thus, there are a whole range of languages based on closed curves that find applications in, or related to, software engineering. A collection of closed curves can be viewed as an Euler diagram but there are a host of differing views of what constitutes such a diagram which will be discussed later in the paper. In all of the application areas mentioned above, the automatic layout of diagrams has the potential to be important and in some cases could be considered essential. For example, when using constraint diagrams to specify operations it might be necessary to prove that the postcondition of one operation implies the precondition of another. In such situations the reasoning may be automated; see [SMF+07]. To display the results of that reasoning to the software engineer, the automatic generation of the diagrams in the proof is needed.

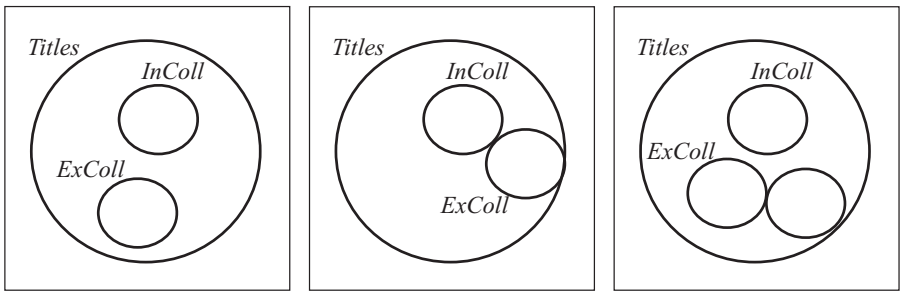


Figure 2: Three Euler diagrams.

Euler diagrams are used to make statements about sets. For example, given the sets $Titles = \{x, y, z\}$, $InColl = \{x, y\}$ and $ExColl = \{z\}$, we can draw any one of the Euler diagrams in figure 2 to represent this situation where the label $Titles$ represents the set $Titles$ and so forth. In general,

the ideal is that an Euler diagram can be found that represents the given collection of sets and has nice visual qualities, allowing the diagram to be interpreted by the user with ease; in figure 2, the lefthand diagram would probably be considered to be the nicest of the three because one can easily differentiate the curves. To this end, most of the generation algorithms developed to date draw Euler diagrams that have certain properties, sometimes called well-formedness conditions. In general, the generation problem is to find an Euler diagram that possess some given properties and represents some specified collection of sets. So, the generation problem can be broken down into a collection of subproblems, one for each set of properties.

Various methods for generating Euler diagrams have been developed, each concentrating on a particular class of Euler diagrams; see, for example [CR05b, CR03, FH02, KMGB05, VV04]. Ideally, such generation algorithms will produce diagrams with desirable properties in an efficient way. The generation algorithms developed so far produce Euler diagrams that have certain sets of properties. However, within the generation community, there is not a commonly agreed definition of an Euler diagram and the properties imposed on them are rarely stated precisely. Such precision is vital if we are to fully explore how to generate Euler diagrams effectively and to answer various related questions such as whether an Euler diagram exists that represents a certain statement and possesses certain properties. In section 2, we provide a general definition of an Euler diagram and formalize various concepts which are necessary for the remainder of the paper. Section 3 provides a formalization of a variety of properties that have been imposed on Euler diagrams, either in existing layout work or in related application areas. Finally, section 4 concludes and discusses implications for layout that imposing these properties brings.

2 Euler Diagram Syntax

There are various definitions of Euler diagrams, most of which start with a set of closed curves and proceed to state that these curves have certain properties. For example, in [FH02] an Euler diagram is a finite set of simple, labelled, closed curves, no pair of which have the same label or meet tangentially, and no point in $\mathbb{R}^2$ is in the image of more than two curves (that is, there are no triple points). By contrast, [VV04] allows curves to have the same label. Exceptions to the curved based definitions include that in [LP97], where an Euler diagram is viewed as a set of connected regions in the plane. In this section, we give a very generic definition of an Euler diagram and build up the necessary language required to formulate certain properties that Euler diagrams may possess. To begin, we recollect the definition of a closed curve.

Definition 1 A **curve** is a continuous function defined on an interval $[x,y] \subset \mathbb{R}$. Let c be a curve with domain $[x,y]$. If $c(x) = c(y)$ then c is **closed** [Bla83].

From this point, we assume that all curves are closed, have domain $[0,1]$ and have codomain $\mathbb{R}^2$ unless stated otherwise. We further assume that all curves $c\colon [x,y] \to \mathbb{R}^2$ can be defined by $c(t) = (f(t),g(t))$ where f and g are continuous; this allows us to utilize various topological concepts necessary for the formalization of the properties Euler diagrams may possess. In an Euler diagram, all curves have a label; we assume that their labels are drawn from a fixed set $\mathscr{L}$.

Definition 2 An Euler diagram, d, is a pair, $(Curve, l)$ where

1. *Curve* is a finite collection of closed curves each with codomain $\mathbb{R}^2$,

2. l: *Curve* $\rightarrow \mathscr{L}$ is a function that returns the label of each curve.

In our figures, we sometimes draw a rectangle around Euler diagrams to delineate them; such rectangles are not formally part of the diagram. Given an Euler diagram, we need to be able to identify the interior of the curves, since these diagrams make statements about set inclusion and disjointness by representing these relations in an isomorphic way; for example, to express *InColl* is a subset of *Titles*, a curve labelled *InColl* is placed *interior* to a curve labelled *Titles*; see figure 2. Given a curve that does not self-intersect, one can easily identify the interior: by the Jordan Curve Theorem, such a curve splits $\mathbb{R}^2$ into two pieces, the bounded piece which forms the interior and the infinite face which is exterior. However, the case for non-simple closed curves is not so obvious; see [FS06] for a discussion on this issue. In the case of any simple closed curve, c, we observe that any point, $p \in \mathbb{R}^2$, is inside c whenever c 'winds' exactly once around that point. For example, in figure 3, the leftmost curve is simple and the point p lies in its interior; here, $c(0)$ denotes where we 'start drawing the curve' and the arrows indicate how we traverse the curve. For all three curves, the point p is interior and the curve winds around p exactly once; the shaded regions represent the interiors of the curves. In the middle and righthand curves, the point q is in the exterior; the middle curve does not wind around q at all whereas the righthand (non-simple) curve winds exactly twice around q. We use this insight to formally define the interior (and exterior) of general closed curves.

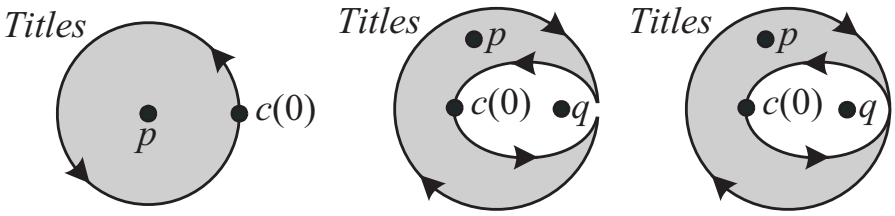


Figure 3: Identifying the interior: winding numbers.

Definition 3 Let c be a closed curve and let p be a point in $\mathbb{R}^2 - im(c)$. The point p is **interior** to c if the winding number of c with respect to p, denoted $wind(c,p)$ is odd, with the set of all such points denoted $int(c)$. All points in $\mathbb{R}^2$ that are not interior to c are **exterior** to c, with the set of all such points denoted $ext(c)$.

For any simple closed curve the winding number around any point in the bounded face is one, thus the definition of interior just given agrees with the intuitive notion interior in such cases. Given an Euler diagram, there may be more than one curve with a given label, such as d_1 in figure 4. In [VV04], an Euler diagram is permitted to have curves with common labels, augmented with a $+$ or a $-$. The diagram d_2 in figure 4 shows such a diagram. There are two curves labelled *Titles* of which one is 'positive', labelled *Titles*$+$, and the other is 'negative', labelled *Titles*$-$. The area inside the curve labelled *Titles*$-$ represents the set *InColl* $-$ *Titles*.

In [VV04], any curve, c_1, with the same label as another curve, c_2, is only placed inside c_2 as

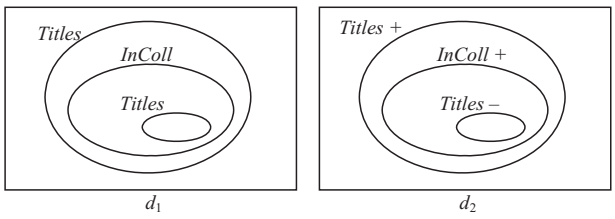


Figure 4: Diagrams with common labels.

a result of their generation algorithm if c_2 is augmented with a plus and c_1 is augmented with minus. As a consequence, the augmentation is redundant because the placement of the curves implicitly provides the sign; the diagram d_1 in figure 4 can be considered as having the same meaning d_2.

In [Joh05], an Euler diagram may also contain two curves with the same label. In general, curves with the same label are referring to the same set. Just as with closed curves, we need to be able to identify the interior and exterior of curves with the same label. For example, d_1 in figure 4 contains two labels: $Titles$, which labels two curves, and $InColl$. The 'interior' of $Titles$ is the set of points interior to the curve labelled $Titles$ which contains the other two curves but not those points interior to the other curve labelled $Titles$. This notion is consistent with our definition of interior for curves, using winding numbers.

Definition 4 Let $(Curve, l)$ be an Euler diagram and let $Cur(L)$ be the set of all curves in d with the label L. A point p is **interior** to $Cur(L)$ if the sum of the winding numbers of the curves in $Cur(L)$ with respect to p is odd; more formally, the sum $\sum_{c \in Cur(L)} wind(c, p)$ is odd. The set of interior points is denoted $int(Cur(L))$. All points in $\mathbb{R}^2$ which are not interior to $Cur(L)$ are **exterior** to $Cur(L)$, the set of which is denoted $ext(Cur(L))$.

For any singleton set of curves of the form $Cur(L)$, we immediately see that the points interior to $Cur(L)$ are the same as those interior to the single curve which $Cur(L)$ contains. A set of curves $Cur(L)$ represents a set and the spatial arrangement of the curves in a diagram provides information about the relationship between those sets. Thus it is useful to identify how the sets of curves of the form $Cur(L)$ partition the plane into pieces.

Definition 5 Let $d = (Curve, l)$ be an Euler diagram and let $Cur \subseteq \{Cur(L) : L \in im(l)\}$. If the set

$$z = \bigcap_{Cur(L') \in Cur} int(Cur(L')) \cap \bigcap_{Cur(L') \in \{Cur(L') : L \in im(l)\} - Cur} ext(Cur(L'))$$

is non-empty then z is a **zone** of d, with the set of such zones denoted $Z(d)$.

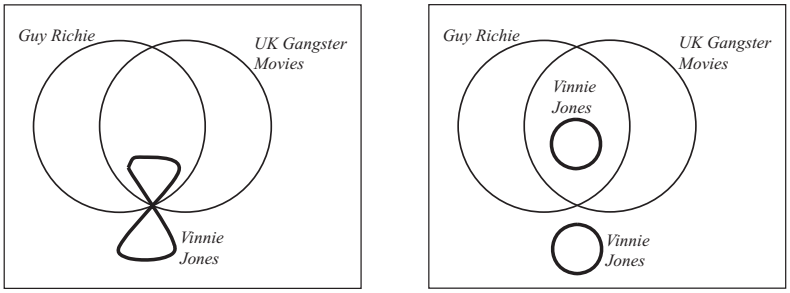
In figure 4, d_1 contains four zones (the four components of $\mathbb{R}^2$ minus the images of the curves). The zones are essentially given rise to by the labels used in the diagram: each subset of the label set gives rise to a set of the form Cur as above.

3 Formalizing Properties

Typically, generation techniques for Euler diagrams aim to produce a layout that possess certain properties deemed desirable, possibly related to the understandability of the resulting diagram or appropriate for the application area. For example, a diagram with a certain property may be more understandable than one without that property. For each of the properties below, we provide examples to motivate their usefulness and discuss where they have been used in the related literature.

3.1 Simplicity Property

Many definitions of Euler (based) diagrams stipulate that the curves must be simple [BR98, CR05b, CR03, Ham95, HST05, SA04, VV04] but others do not [BH96, CC04, DC05, SK00, Shi94, SA05]. Indeed, the layout work in [CR05b, FH02, VV04] assumes that all curves are simple. To recall, a simple curve is one that does not self-intersect. In figure 5, there are two diagrams with the same meaning and the lefthand side one contains a non-simple curve, labelled *Vinnie Jones*.



A diagram with a non simple curve. A diagram with a duplicated curve label.

Figure 5: Simplicity of curves.

Definition 6 A curve c is **simple** if for all $x, y \in [0,1]$, $c(x) = c(y)$ implies $x = y$ or $|x - y| = 1$.

Definition 7 An Euler diagram, d, possesses the **simplicity property** if and only if all of the curves in d are simple.

Euler diagrams that possess the simplicity property are highly desirable due to the potential difficulty users may meet when interpreting diagrams whose curves are not simple.

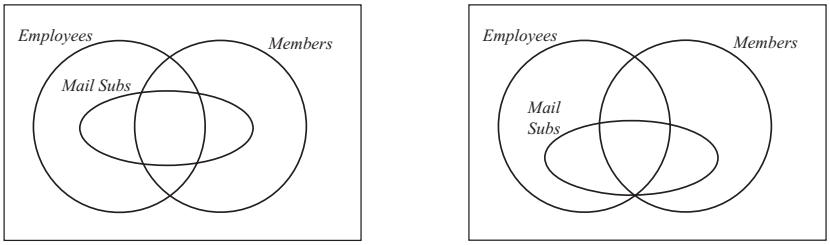
3.2 Unique Labelling Property

The definitions of an Euler diagram given in [BH96, BR98, CC04, CR05b, CR03, DC05, Ham95, HST05, SK00, SA05, SA04] allow labels to occur at most once in any Euler diagram. Thus, the *unique labelling* property is one that is commonly imposed on Euler diagrams. An example can be seen in figure 5, where the lefthand diagram only uses labels once whereas the righthand diagram uses *Vinny Jones* twice.

Definition 8 An Euler diagram, $(Curve, l)$, possesses the **unique labelling property** if the function l is injective.

3.3 Connected Zones Property

The lefthand diagram in figure 6 contains a zone (that inside both *Employees* and *Members*) which consists of two parts. Such a zone is said to be *disconnected*. Frequently, zones are required to form connected components of $\mathbb{R}^2$, as in [FH02, VV04, CR05b]. It can be hard to interpret Euler diagrams with disconnected zones under some circumstances, for example when items are placed in zones. Such a collection of items might be split up in two or more *minimal regions*, thus not allowing an immediate interpretation about the number or distribution of items.



A diagram with a disconnected zone. A diagram with a 3 -point (triple point).

Figure 6: Disconnected zones and multiple points.

Definition 9 Let $d = (Curve, l)$ be an Euler diagram. A non-empty set of points, m, in $\mathbb{R}^2$ is a **minimal region** of d provided m is a connected component of $\mathbb{R}^2 - \bigcup_{c \in Curve} im(c)$ with the set of such minimal regions denoted $M(d)$.

Definition 10 An Euler diagram, d, possesses the **connected zones property** if all of the zones of d are also minimal regions of d, that is $Z(d) = M(d)$.

3.4 Zone Area Property

It is often desirable to visualize some value associated with the zones such as the cardinalities of the sets they represent. This motivates the need for generating *area proportional* Euler diagrams, which are characterized by using a weight function to measure zone area. Thus we introduce a *zone area* property that Euler diagrams may possess. To satisfy this property, the zones in an Euler diagram must have the area specified by a weight function. As an example, figure 7 shows an area proportional diagram providing information about the number of films in different categories. The weight function does not assign an area to the zone which is not inside any curves. In figure 7 the zone sizes are proportional to the weights within them.

Definition 11 Let d be an Euler diagram and

$$w\colon Z(d) - \{ \bigcap_{Cur(L') \in \{(Cur(L)) : L \in im(l)\}} ext(Cur(L'))\} \to \mathbb{R}^+$$

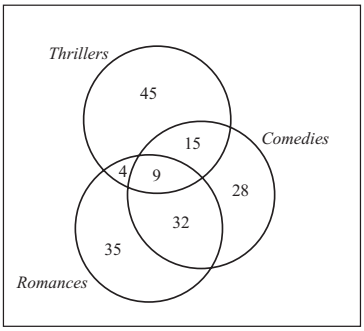


Figure 7: An area proportional diagram.

be a function. The diagram d possesses the ***w*-zone area property** if and only if all of the zones, z in the domain of w have area $w(z)$.

There are obvious variations of this property; for example, when drawing bounding rectangles around Euler diagrams, rather than embedding them in the whole of $\mathbb{R}^2$, one can stipulate the area of all of the zones.

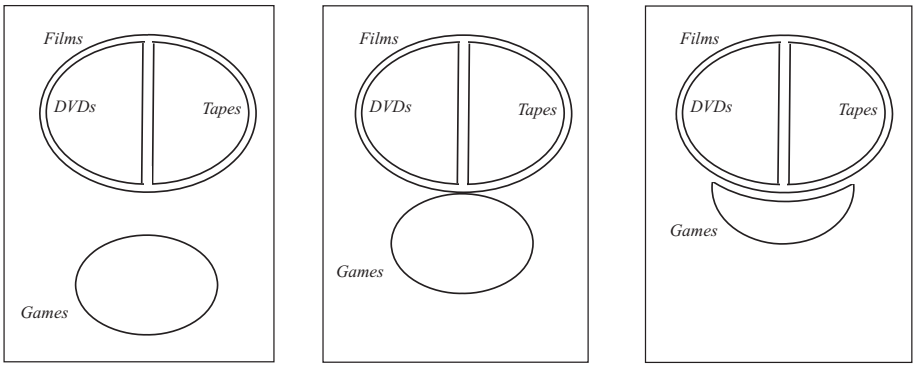
The work on laying out area proportional Euler diagrams typically produces diagrams that have connected zones [CR03, CR05b, KMGB05]. Furthermore, some approaches to layout attempt to find area proportional diagrams with unique labelling and the property that each curve is a circle [CR05a, KMGB05]. Exact area proportional layouts under these conditions is not always possible. The desirability of utilizing circles and getting an approximate result has also led to the notion of relative size, so that more diagrams can be drawn where the size of one zone is specified to be bigger than another [CR05a].

3.5 Connected Diagram Property

A diagram is said to be *connected* if the (images of) the curves form a connected set, such as that in figure 7. Any collection of sets can be represented by a connected diagram: intuitively, any disconnected diagram can be made connected by creating *brushing* points (defined later); [Cho07] uses this result in the generation process. For example, figure 8 shows a disconnected diagram (left) which is 'made connected' by moving the curve labelled *Games* upwards, shown in the middle diagram. All of the diagrams in this figure contain concurrent curves, but they have been slightly pulled apart for visual clarity.

Definition 12 An Euler diagram, $d = (C, l)$, possesses the **connected diagram property** if $\bigcup_{c \in C} im(c)$ is connected.

Given a collection of sets, if we wish to find a diagram representing that collection but do not wish it to possess the connected diagram property then we can embed the disconnected parts separately; it is possible to identify disconnected (*atomic*) parts from the set information [FHT04].



A disconnected diagram. A diagram with two curves brushing. A diagram with partial concurrency.

Figure 8: Three diagrams with concurrent curves.

3.6 Multiple Point Properties

Given an Euler diagram that possesses the simplicity property, a point p in $\mathbb{R}^2$ is called an n-point if the number of curves that pass through p is exactly n. In the non-simple case, curves can pass through a point more than once and in such cases the number of times each curve passes through p contributes to the value of n. For example, the righthand diagram in figure 6 contains a 3-point, also called a triple point. With non-simple curves, a triple point can be formed from a single curve. A diagram in which all points are at most 2-points can make a diagram easier to understand, but reduces the number of diagrams that can be drawn without introducing other undesirable properties, such as concurrency and non-simple curves. The layout method in [FH02] draws diagrams that have a most 2-points.

Definition 13 Let $d = (Curve, l)$ be an Euler diagram and let p be a point in $\mathbb{R}^2$. We say that p is an ***n*-point** in d if $\sum_{c \in Curve} \left|\{x \in [0,1) : c(x) = p\}\right| = n$.

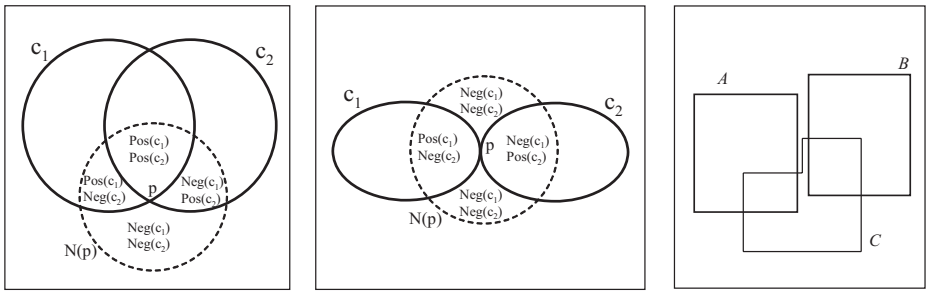
Definition 14 An Euler diagram, $d = (Curve, l)$, possesses the ***n*-point property** provided each point in $\mathbb{R}^2$ is at most an n-point in d.

3.7 Curve Crossing Property

In an Euler diagram, d, two curves in d may cross at a point p. Furthermore, the two curves might intersect at p but not cross, in which case they *brush* at p. It may well be the case that two curves brush and cross at a point. For example, all intersection points of the curves in the diagram in figure 7 are crossing points whereas the middle diagram in figure 8 contains a brushing point, where *Games* and *Films* intersect. It is possible for a point to be both a crossing and a brushing point. We now proceed with a series of definitions that allow us to identify whether two curves cross at a point and whether they brush at a point. Some generation algorithms produce diagrams that only permit curves to intersect only at crossing point that are not brushing points [FH02].

As an example, the curves in the lefthand diagram in figure 9, cross at the point p. To identify this formally, we can consider a disc neighbourhood $N(p)$ around p see that it contains four

regions, one for each of the four combinations of being 'inside' or 'outside' c_1 and c_2. However, had c_1, say, been non-simple it need not have had an interior. Thus, we cannot use the notion of interior to define a crossing point. Given c_1, $N(p)$ is cut into two pieces and we can arbitrarily assign positive and negative to each of these two pieces respectively. The curves in the middle diagram brush at p. Here, the positive and negative regions do not correspond to each of the four combinations of being positive or negative to the two curves.



A diagram with a crossing point. A diagram with a brushing point. A diagram with brushing and crossing concurrency.

Figure 9: Crossing and brushing curves.

Definition 15 Let $c\colon [x,y] \to \mathbb{R}^2$, where the notation $[x,y]$ denotes any closed interval of $\mathbb{R}^2$. Let p be a point in $im(c)$ and let $N(p)$ be a disc neighbourhood of p. If $N(p) - im(c)$ consists of exactly two connected components then $N(p)$ is called a **splitting neighbourhood** of p for c.

Definition 16 Let $c_1\colon [x,y] \to \mathbb{R}^2$ and $c_2\colon [a,b] \to \mathbb{R}^2$ be two curves such that there is a unique point p in $im(c_1) \cap im(c_2)$. If there exists a disc neighbourhood $N(p)$ such that

1. $N(p)$ is a splitting neighbourhood of p for c_1; arbitrarily call one of the two connected components $Pos(c_1)$ and the other $Neg(c_1)$,
2. $N(p)$ is a splitting neighbourhood of p for c_2; arbitrarily call one of the two connected components $Pos(c_2)$ and the other $Neg(c_2)$,
3. $N(p)$ contains a point in $Pos(c_1) \cap Pos(c_2)$,
4. $N(p)$ contains a point in $Pos(c_1) \cap Neg(c_2)$,
5. $N(p)$ contains a point in $Neg(c_1) \cap Pos(c_2)$ and
6. $N(p)$ contains a point in $Neg(c_1) \cap Neg(c_2)$

then c_1 and c_2 are said to **cross** at p. Otherwise c_1 and c_2 **brush** at p.

Let $c_1\colon [0,1] \to \mathbb{R}^2$ and $c_2\colon [0,1] \to \mathbb{R}^2$ be two closed curves such that there is a point p in $im(c_1) \cap im(c_2)$. If there exist closed intervals $I_1, I_2 \subseteq [0,1]$ such that

1. $im(c_1|_{I_1}) \cap im(c_2|_{I_2})$ contains exactly p,
2. the curves $c_1|$ and $c_2|$ cross at p

then c_1 and c_2 **cross** at p. If there exists $I_1, I_2 \subseteq [0,1]$ such that

1. $im(c_1|_{I_1}) \cap im(c_2|_{I_2})$ contains exactly p,
2. the curves $c_1|$ and $c_2|$ brush at p

then c_1 and c_2 **brush** at p.

Definition 17 An Euler diagram, d, possesses the **crossing property** if and only if whenever two curves, C_1 and C_2, in d intersect they cross but do not brush.

The concepts of crossing and brushing can be generalized to curve segments as well as being defined for points; see the discrete property below.

3.8 Discrete Property

The embedding methods of [VV04, CR05b] sometimes produce diagrams with concurrent line segments. If concurrency is permitted, along with either non-simple curves or multiple curves with the same label, then any collection of sets can be represented by an Euler diagram. Moreover, many collections of sets can only be represented by Euler diagrams with concurrent line segments when certain properties are imposed. However, concurrent line segments can make it difficult to interpret a diagram; the generation algorithm in [FH02] only produces diagrams without concurrency; thus the curves in these diagrams intersect only at a discrete set of points.

Definition 18 A set of points, $X \subseteq \mathbb{R}^2$, is **discrete** if for every point $x \in X$ there exists $\varepsilon > 0$ such that

$$\{p \in \mathbb{R}^2 : d(p,x) \leq \varepsilon\} \cap X = \{x\}$$

where $d(p,x)$ is the standard Euclidean distance between p and x.

Definition 19 An Euler diagram, $d = (Curve, l)$, possesses the **discrete property** if all pairs of curves in $Curve$ do not run concurrently:

$$\{x \in \mathbb{R}^2 : \exists a, b \in [0,1] a \neq b \wedge c_1(a) = c_2(b) = x\}$$

is a discrete set of points.

So, any diagram that possesses the discrete property does not have concurrent curves. With non-simple curves, self concurrency is possible, so that the curve is concurrent with itself.

There are a number of variations of concurrency. Its possible to have 'crossing' or 'brushing' concurrency depending on whether the curves cross at the start and end of the concurrency segment. For example, the righthand diagram in figure 9 A and B brush concurrently (where the concurrent line segments have been drawn slightly pulled apart for clarity) and A crosses C concurrently, as does B.

An important variation is the distinction between 'full' concurrency and 'partial' concurrency. Full concurrency can be seen in figure 8, with the curves labelled *DVDs* and *Films* have full concurreny, as do *Tapes* and *Films* whereas the curves labelled *Games* and *Films* have partial concurrency.

For full concurrency the line segments of the two curves, c_1 and c_2, separate at a point which is a crossing point or a brushing point for some pair of curves or a point where another curve separates from a concurrent line segment; any other concurrency is partial. It is thought that partial concurrency can always be removed from a diagram.

3.9 Curve Shape Properties

Curves may have the property of having a particular geometric shape, such as circle, oval, triangle, square or being n-gons; the work in [CR05a, KMGB05] generates Euler diagrams consisting only of circles and that in [CFW05] is concerned with k-gons with a particular focus on triangles. A weaker constraint may be to require that all curves are the same shape, even if it is not regular. Further, a curve may be smooth, in the sense that its function is differentiable. Sometimes there is a requirement that all curves are convex, so concave curves are not permitted, see, for example, [LP97]. In practice, most implemented diagram visualization methods restrict the shape of the curves to some extent, due to the restrictions on line visualization in software environments; for example, [CR05a, FH02, VV04] generates diagrams whose curves are polygons.

Definition 20 An Euler diagram, $d = (Curve, l)$, possesses the **smooth property** if all of the curves in $Curve$ are smooth.

A curve, c, is **convex** if it is simple and its interior is convex; that is, for each pair of points in the interior of c there is a straight line between those points that lies in the interior of c.

Definition 21 An Euler diagram, $d = (Curve, l)$, possesses the **convex property** if all of the curves in $Curve$ are convex.

Definition 22 Let S be a set of shapes. An Euler diagram, $d = (Curve, l)$, possesses the **S-shapes property** if all of the curves in $Curve$ are one of the shapes in S.

4 Conclusion

In this paper, we have provided a very general definition of an Euler diagram and provided a formalization of many properties that such diagrams are frequently required to possess. This definition and formalization provides a common language for the Euler diagram community to utilize. A consequence of using this common language will be more accessible and better integrated research results. Given the wide variety of languages based on closed curves that are utilized in software engineering, the framework described here has the potential to bring many important benefits.

There is a range of further properties that Euler diagrams may possess that can also be considered desirable which we have not formalized in this paper. However, they are not so commonly considered by Euler diagram users. A *monotonic diagram* is one which always allows zones to be resized or collapsed and still maintain simple curves. It is a property of diagrams formed from the [CR05b] embedding process, which produces an area proportional diagram from a Venn diagram construction. We do not include it amongst the main properties, as the definition of a

monotonic diagram relies on the notion of the *dual* of the Euler diagram, rather than the direct definition of a diagram. It is also possible to add zones to diagrams in order to ensure a diagram has certain properties. Here, the added zones might be shaded, as with the original work of Venn [Ven80] and the embedding process of [KMGB05] which does not guarantee that the initially specified zones are present. In this case a desirable property of a diagram might be that it has the smallest number of extra zones so as to possess certain other properties. One area of study in the Venn community is the notion of drawing a diagram in a symmetric manner. Although certain collections of sets can be represented by symmetric Euler diagrams, in the general case symmetry is not possible. It is also unclear how much an aid to user interpretation of a diagram is helped by symmetry. The context in which an Euler diagram is to be used is likely to influence those properties deemed desirable.

The generation work to date produces diagrams that have specified selections of properties we have formalized. Many difficult open problems remain to be solved. Given a collection of sets, it is unknown which properties can be possessed by Euler diagrams which represent those collections. For instance, classifying exactly which collections of sets can be represented by diagrams which possess the simplicity and unique labelling properties remains unanswered. We might also choose which properties a generated Euler diagram has, given a collection of sets that we wish to visualize. An algorithm that incorporates such user choice where possible has not yet been developed. Some collections of sets can be represented by an Euler diagram that possess exactly one of the discrete property and the 2-point property; under such circumstances these two properties are *exchangeable*. Further study into which properties can be exchanged needs to be performed.

Acknowledgements: This work is support by ESPRC grants EP/E011160/1 and EP/E010393/1 for the Visualization with Euler Diagrams project. Additionally, Gem Stapleton is supported by a Leverhulme Trust Early Career Fellowship. Thanks to Andrew Fish for discussing various aspects of this paper.

Bibliography

[BH96] J. Barwise, E. Hammer. Diagrams and the Concept of Logical System. In Allwein and Barwise (eds.). Oxford University Press, 1996.

[Bla83] D. Blackett. *Elementary Topology*. Academic Press, 1983.

[BR98] B. Bultena, F. Ruskey. Venn Diagrams with Few Vertices. *Electronic Journal of Combinatorics* 5:1–21, 1998.

[CC04] L. Choudhury, M. K. Chakraborty. On Extending Venn Diagrams by Augmenting Names of Individuals. *Proceedings of 3rd International Conference on the Theory and Application of Diagrams*, Springer, pages 142–146, March 2004.

[CFW05] J. Carroll, F. Ruskey, M. Weston. Which n-Venn Diagrams can be Drawn with k-gons. *Proceedings of Euler Diagrams 2005*, 2005.

[Cho07] S. Chow. *Generating and Drawing Area-Proportional Euler and Venn Diagrams*. PhD thesis, University of Victoria, 2007.

[Cla05] R. Clark. Failure Mode Modular De-Composition Using Spider Diagrams. *Proceedings of Euler Diagrams 2004* Elsevier, ENTCS vol. 134, pages 19–31, 2005.

[CR03] S. Chow, F. Ruskey. Drawing Area-Proportional Venn and Euler Diagrams. *Proceedings of Graph Drawing 2003, Perugia, Italy*, Springer, 466–477, September 2003.

[CR05a] S. Chow, P. Rodgers. Constructing Area-Proportional Venn and Euler Diagrams with Three Circles. *Proceedings of Euler Diagrams 2005*, 2005.

[CR05b] S. Chow, F. Ruskey. Towards a General Solution to Drawing Area-Proportional Euler Diagrams. *Proceedings of Euler Diagrams*, Elsevier, ENTCS vol 134, pages 3–18, 2005.

[DC05] H. Dunn-Davies, R. Cunningham. Propostional Statecharts for Agent Interaction Protocols. *Proceedings of Euler Diagrams 2004, Brighton, UK* Elsevier, ENTCS vol 134, pages 55–75, 2005.

[DES03] R. DeChiara, U. Erra, V. Scarano. A System for Virtual Directories Using Euler Diagrams. Proceedings of Information Visualisation, IEEE Computer Society, pages 120-126, 2003.

[DS07] A. Delaney, G. Stapleton. On the Descriptional Complexity of a Diagrammatic Notation. *Accepted for Visual Languages and Computing*, Knowledge Systems Institute, 2007.

[FH02] J. Flower, J. Howse. Generating Euler Diagrams. *Proceedings of 2nd International Conference on the Theory and Application of Diagrams*, Springer, pages 61–75, April 2002.

[FHT04] J. Flower, J. Howse, J. Taylor. Nesting in Euler diagrams: syntax, semantics and construction. *Software and Systems Modelling* 3:55–67, March 2004.

[FS06] A. Fish, G. Stapleton. Formal Issues in Languages Based on Closed Curves. *Proceedings of Distributed Multimedia Systems, International Workshop on Visual Languages and Computings*, Knowledge Systems Institute, pages 161–167, 2006.

[Ham95] E. Hammer. *Logic and Visual Information*. CSLI Publications, 1995.

[HES+05] P. Hayes, T. Eskridge, R. Saavedra, T. Reichherzer, M. Mehrotra, D. Bobrovnikoff. Collaborative Knowledge Capture in Ontologies. *Proceedings of the 3rd International Conference on Knowledge Capture*, pp. 99–106, 2005.

[HS05] J. Howse, S. Schuman. Precise Visual Modelling. *Journal of Software and Systems Modeling* 4:310–325, 2005.

[HST05] J. Howse, G. Stapleton, J. Taylor. Spider Diagrams. *LMS Journal of Computation and Mathematics* 8:145–194, 2005.

[Joh05] C. John. Projected Contours in Euler Diagrams. *Euler Diagrams 2004* Elsevier, ENTCS vol 134, pages 103–126, 2005.

[KC99] S.-K. Kim, D. Carrington. Visualization of Formal Specifications. *6th Asia Pacific Software Engineering Conference*, IEEE Computer Society Press, pages 102–109, 1999.

[Ken97] S. Kent. Constraint Diagrams: Visualizing Invariants in Object Oriented Modelling. *Proceedings of OOPSLA97*, pages 327–341, October 1997.

[KMGB05] 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.

[Lov02] J. Lovdahl. *Towards a Visual Editing Environment for the Languages of the Semantic Web*. PhD thesis, Linkoping University, 2002.

[LP97] O. Lemon, I. Pratt. Spatial Logic and the Complexity of Diagrammatic Reasoning. *Machine GRAPHICS and VISION* 6(1):89–108, 1997.

[SA04] N. Swoboda, 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.

[SA05] N. Swoboda, G. Allwein. Heterogeneous Reasoning with Euler/Venn Diagrams Containing Named Constants and FOL. *Proceedings of Euler Diagrams 2004* Elsevier, ENTCS vol 134, 2005.

[Shi94] S.-J. Shin. *The Logical Status of Diagrams*. Cambridge University Press, 1994.

[SK00] H. Sawamura, K. Kiyozuka. JVenn: A Visual Reasoning System with Diagrams and Sentences. *Proceedings of 1st International Conference on the Theory and Application of Diagrams* Springer, pages 271–285, 2000.

[SMF+07] G. Stapleton, J. Masthoff, J. Flower, A. Fish, J. Southern. Automated Theorem Proving in Euler Diagrams Systems. *Journal of Automated Reasoning*, June 2007.

[TVV05] J. Thiévre, M. Viaud, A. Verroust-Blondet. Using Euler Diagrams in Traditional Library Environments. *Euler Diagrams 2004* Elsevier, ENTCS vol 134, pages 189–202, 2005.

[Ven80] J. Venn. On the diagrammatic and mechanical representation of propositions and reasonings, Phil.Mag, 1880.

[VV04] A. Verroust, M.-L. Viaud. Ensuring the Drawability of Euler Diagrams for up to Eight Sets. *Proceedings of 3rd International Conference on the Theory and Application of Diagrams* Springer, pages 128–141, 2004.