Electronic Communications of the EASST
Volume 13 (2008)

Proceedings of the Second International Workshop on Layout of (Software) Engineering Diagrams (LED 2008)

Generating Euler Diagrams from Existing Layouts

Gem Stapleton, John Howse Peter Rodgers and Leishi Zhang

16 pages

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

Generating Euler Diagrams from Existing Layouts

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

[1] g.e.stapleton@brighton.ac.uk,[2] john.howse@brighton.ac.uk
www.cmis.brighton.ac.uk/reseach/vmg
University of Brighton, UK

[3] p.j.rodgers@kent.ac.uk, [4] l.zhang@kent.ac.uk
University of Kent, UK

Abstract: Euler diagrams have a wide variety of uses, from information visualization to logical reasoning. In the case of software engineering, they form the basis of a number of notations, such as state charts and constraint diagrams. In all of their application areas, the ability to automatically layout Euler diagrams brings considerable benefits. There have been several recent contributions towards the automatic generation and layout of Euler diagrams, all of which start from an abstract description of the diagram and produce a collection of closed curves embedded in the plane. In this paper, we are concerned with producing layouts by modifying existing ones. This type of layout approach is particularly useful in domains where we require an updated, or modified, diagram such as in a logical reasoning context. We provide two methods to add a curve to an Euler diagram in order to create a new diagram. The first method is guaranteed to produce layouts that meet specified well-formedness conditions that are typically chosen by others who produced generation algorithms; these conditions are thought to correlate well accurate user interpretation. We also overview a second method that can be used to produce a layout of any abstract description.

Keywords: Information visualization, diagram layout, Venn diagrams

1 Introduction

Automated diagram layout has the potential to bring huge benefits and it is unsurprising that, with the computing power now available, considerable research effort is focused on this topic. In software engineering, the prevalent use of diagrammatic notations makes this area an ideal candidate to benefit from state-of-the-art generation and layout techniques. Many diagrams are based on collections of closed (usually simple) curves, such as state charts and class diagrams both of which are part of the array of languages that form the UML. Various other languages are based on closed curves, such as constraint diagrams, that are designed for software specification; see [HS05, KC99] for examples of such specifications. To illustrate, the constraint diagram in figure 1 expresses that every store stocks at least two copies of some film in its collection. A well studied fragment of the constraint diagram language, called spider diagrams, that is also based on closed curves has been used in a variety of application areas; see, for example, [Cla05, Nie2006].

A finite collection of closed curves constitutes an Euler diagram and, therefore, the languages

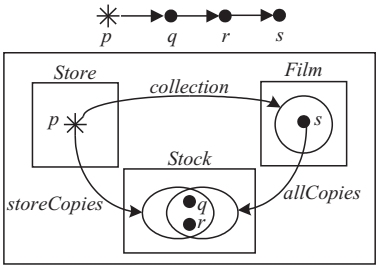


Figure 1: A constraint diagram.

mentioned above can all be viewed as extending Euler diagrams in some manner. Thus, the automated layout of Euler diagrams provides an essential basis for the automated layout of a large range of other diagrams. In addition to those mentioned above, Euler diagrams have numerous other application areas; for example [DES03, HES+05, KMG+05, Lov02, TVV05].

Various methods for generating Euler diagrams have been developed, each concentrating on a particular class of Euler diagrams; see, for example [CR05b, CR03, FH02, KMG+05, RZF08, VV04]. Ideally, such generation algorithms will produce diagrams with desirable properties in an efficient way; such properties are sometimes called wellformedness conditions and will be more fully explained below. The generation algorithms developed so far produce Euler diagrams that have certain sets of properties. Each of these generation methods starts with an abstract description of the required diagram and proceeds to seek a layout.

In this paper, we take a different approach to generation, in that we take an existing diagram layout and transform it in to another layout. In particular, we describe how to add a curve to an existing layout to create a new Euler diagram using two methods. The first method is presented in two stages, with the first stage describing how to add a new curve to a so-called wellformed layout in such a way that each 'minimal region' is split in to two minimal regions (one inside and one outside the new curve) and wellformedness is maintained. The technique is extended to allow selected minimal regions to be split, others to be completely contained by the new curve and the rest to be completely outside the new curve. In fact, our technique guarantees to be able to find an embedding of the new curve in the required manner whenever this is possible given the existing layout. The second method (informally outlined in the paper) can be used to find a layout of any Euler diagram description; we can decompose the layout problem in to a sequence of layout problems, where we add a new curve at each stage. Thus, in this paper we provide two approaches to Euler diagram generation that, in addition to contributing to the general generation problem, are particularly advantageous in any situation where we wish to modify a diagram by adding a curve and maintain the existing layout.

In section 2, we provide motivation specific to our generation approach of adding a curve and section 3 overviews the syntax of Euler diagrams and other necessary background material. Section 4 defines the operation of adding a curve to Euler diagrams and their descriptions. In section 5 we show how to add a curve to a so-called atomic wellformed Euler diagram and prove that the resulting diagram is also atomic and wellformed. Section 6 generalizes to the non-atomic (nested) case. Section 7 shows how to add a curve to an arbitrary Euler diagrams; the technique

ensures that any abstract Euler diagram can be generated by adding curves one at a time. The application of our layout technique to the general generation problem is discussed in section 8. Finally, section 9 discusses a prototype implementation of the approach and presents some output from the software.

2 Motivation

Euler diagram generation is hard and, as the number of curves increases, the layout problem becomes increasingly more difficult. Moreover, we may have added requirements of our layouts in certain contexts.

For example, Euler diagrams often form the basis of visual logics and, in such settings, the automated layout of diagrams is essential when building theorem provers. A common operation in reasoning systems based on Euler diagrams is to add a curve to a diagram in such a manner that each so-called zone (defined later) splits in to two new regions, one inside and the other outside the new curve [SK00, Shi94, SA04]. To illustrate, in figure 2, we can add a curve to d_1 to give d_2. The diagram d_2 has the same abstract description as d_3 but looks rather different. At the abstract level, adding a curve to $ab(d_1)$, the abstract description of d_1, would give $ab(d_2) = ab(d_3)$. If we want to preserve the layout of d_1 when adding the curve, we need some method to generate d_2, rather than go via the abstract syntax, $d_1 \mapsto ab(d_1) \mapsto ab(d_2)$, and then generate a concrete diagram with abstraction $ab(d_2)$, which could result in d_3. Moreover, sometimes we want to

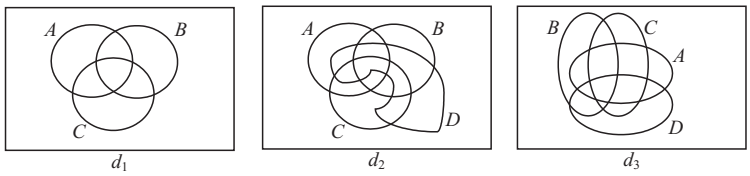


Figure 2: Adding a curve.

add a curve to an Euler diagram in such a manner that not every zone is split in two, such as in [SMF+07]. In this, and other areas, it can be helpful to layout one diagram so that it looks similar to another, preserving as much of the user's mental map as possible [MEL+95]. Our approach to layout gives this preservation for free, in that we take an existing layout and add to it a curve.

Another area where our approach to generation will be particularly helpful is when we utilize a library of nicely drawn examples (such as with circles) as a basis for producing further layouts; see [SFR07] for preliminary work towards building such a library. For example, such a library might include a layout for each abstract description where at most three curves are used. If we then wanted a layout of a diagram containing four curves, we can extract from the library a good layout of an appropriate three curve diagram and add the forth curve to produce the required diagram.

3 Euler Diagrams

We now overview a formalization of Euler diagrams and their descriptions. Moreover, we also describe various concepts that will be required throughout the paper, in particular *wellformed* and *atomic* diagrams.

3.1 Concrete Diagrams

As stated above, an Euler diagram is a collection of closed curves drawn in the plane. We assume that each curve has a label chosen from some fixed set of labels, $\mathscr{L}$.

Definition 1 A concrete Euler diagram is a pair, $d = (Curve, l)$, 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.

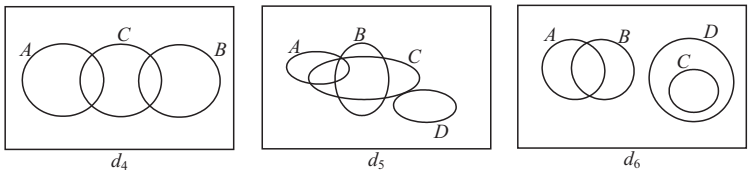


Figure 3: Concrete Euler diagram syntax.

For example, d_4 in figure 3 contains three curves labelled A, B and C. To be more precise, d_4 depicts the images of three simple closed curves. Given a curve, $c\colon [0,1] \rightarrow \mathbb{R}^2$ say, we denote the image of c by $im(c)$ (following the standard notation for the image of a function). The curves partition $\mathbb{R}^2 - \bigcup_{c \in Curve} im(c)$ into connected regions of the plane, called **minimal regions**. A **contour** in an Euler diagram is the set of curves in that diagram with the same label. A point is interior to a contour if it is inside an odd number of its curves, otherwise it is exterior. For formal definitions of the interior of curves in the non-simple case see [SRH+07]. A **zone** in a diagram is a maximal set of minimal regions that can be described as being inside certain contours (possibly no contours) and exterior to the remaining contours. In figure 3, d_4 has six zones, of which two are inside A. The diagram d_5 has ten minimal regions but only eight zones, such as the disconnected zone inside B but outside the remaining curves.

Concrete Euler diagrams may possess certain properties, sometimes called wellformedness conditions, such as containing no triple points (where three or more curves intersect at a single point) or no concurrency between curves (where curves intersect at a non-discrete set of points). Typically, generation algorithms produce concrete diagrams that possess certain properties, in part for reasons of interpretability. We say that a concrete Euler diagram, $(Curve, l)$ is **wellformed** if

1. the function l is injective (no pair of distinct curves have the same label),
2. all of the curves are simple (no curve self-intersects),

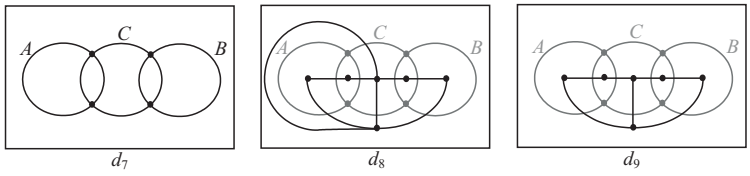
3. there are no triple points of intersection between curves,
4. the zones are connected (each zone consists of exactly one minimal region), and
5. every time two curves intersect they do so transversely (note that this implies that no curves run concurrently);

see [SRH+07] for formalizations of these properties. The generation algorithm in [FH02], for example, draws diagrams that are wellformed.

In figure 3, both d_4 and d_6 are wellformed but d_5 is not. Whilst all of d_5's curves are simple (that is, they do not self-intersect), it has a triple point where A, B and C intersect, the zones are not all connected, and the curves C and D do not meet transversely at the point they intersect.

The concept of *nesting* in diagrams is of particular importance in automated layout. The (images of the) curves in a concrete Euler diagram form connected components of $\mathbb{R}^2$. If the curves give rise to exactly one connected component then the diagram is called **atomic**, otherwise the diagram is **nested** [FHT04]. In figure 3, d_4 and d_5 are atomic whereas d_6 is nested and comprises three atomic components. When laying out nested diagrams, we can automatically generate each of the atomic components separately and then merge the results together. In the case of wellformed diagrams, it has been shown that nestedness can be detected from diagram descriptions and the atomic components identified prior to layout [FHT04].

Concrete Euler diagrams are associated with various graphs. These dual graphs play an instrumental role in their automated layout; see [Cho07, FH02] for more details. First, we can take a concrete diagram, $d_1 = (Curve, l)$ and construct its *Euler graph* which has a vertex at each point two curves meet, these vertices are shown in d_7, figure 4, and the edges are then the curve segments that connect the vertices. As a special case, the Euler graph of a diagram containing a single curve has exactly one vertex placed on that curve. From the Euler graph, we can construct an *Euler graph dual* which is simply a dual graph of the Euler graph, as shown in d_8. Finally, we have what is called a *concrete dual*. A concrete dual graph is a maximal subgraph of an Euler graph dual that contains all of the vertices but no multiple edges. The diagram d_9 shows one concrete dual that can be derived from the depicted Euler dual. We note that all of the graphs described include layout information and are plane. For the purposes of this paper, we require



d_7 d_8 d_9

Figure 4: Various graphs.

the following results.

Lemma 1 *Let d be a concrete Euler diagram. Any two concrete dual graphs (duals of Euler graphs) of d, are isomorphic.*

Corollary 1 *Let d be a concrete Euler diagram. Given any two concrete dual graphs (duals of Euler graphs) of d they are either both Hamiltonian or neither are Hamiltonian.*

In the context of layout, typical algorithms take a diagram description, convert it into an abstract dual graph (sometimes called the superdual) and subsequently seek subgraphs of this abstract dual as candidate concrete dual graphs. Once an appropriate candidate concrete dual graph is found, the curves are embedded around the vertices.

3.2 Diagram Descriptions

In order to generate an Euler diagram, we start with a description of that diagram. To illustrate, d_4 in figure 3 can be described as having three curves, A, B and C, which are also contours. These contours divide the plane in such a manner that there are six zones present. Each zone can be described as being inside certain contours and outside the remaining contours. For instance, there is one zone inside A only and another zone inside precisely A and C. Thus, each present zone can be described by the labels of the contours that the zone is inside. Note that there is always a zone outside all of the contours (the infinite minimal region).

Definition 2 An **abstract Euler diagram description** (or, simply, abstract description), d, is a pair, (l,Z) where $l = l(d)$ is a subset of $\mathscr{L}$ and $Z = Z(d) \subseteq \mathbb{P}l$ such that $\emptyset \in Z(d)$. Elements of Z are called (abstract) **zones**.

Definition 3 Given a concrete Euler diagram $d = (Curve, l)$, we map d to abstract description $ab(d) = (im(l), Z)$, called the **abstraction** of d, where Z contains exactly one abstract zone for each concrete zone in d; in particular, given a concrete zone, z, in d, the abstract zone

$$ab(z) = \{l(c) : c \in C(z)\}$$

is in Z where $C(z)$ is the set of curves in d that contain z.

The diagram d_4 in figure 3 has abstraction (L,Z) where $L = \{A,B,C\}$ and

$$Z = \{\emptyset, \{A\}, \{A,C\}, \{C\}, \{B,C\}, \{B\}\}.$$

The generation problem can be summarized as *given an abstract description, d_1, find a concrete Euler diagram, d_2, such that $ab(d_2) = d_1$*. Sometimes, the generation problem is restricted by imposing certain conditions on d_2, for instance requiring that the concrete diagram is well-formed.

4 Adding a Curve

There are situations when we want to add a curve to a given layout in a specified manner, such as when generating diagrams. In order to do this, we need to know what is meant by adding a curve and how to specify its addition. In this section, we formalize the notion of adding a curve at both the abstract and concrete levels. At the concrete level this is easy: we simply take a diagram and add to it a curve and a label. In figure 5, we add C to d_{10}, giving d_{11}.

Definition 4 Let $d = (Curve, l)$ be a concrete Euler diagram. Let L be a label in $\mathscr{L}$ and let c be a curve that is not in $Curve$. Then we defined $d + (c,L) = (Curve \cup \{c\}, l \cup (c,L))$. If $L \notin im(l)$

and the number of zones doubles when adding c then $d+(c,L)$ is said to be derived from d by splitting each zone.

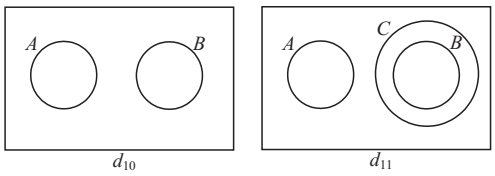


Figure 5: A more complex example.

At the abstract level, things are not quite so straightforward. We note that each zone can be either completely contained by the new curve, completely outside the new curve, or split by the new curve. In figure 5, the diagram d_{11} has a curve C that completely contains the zone inside B, is completely outside the zone inside A and splits the zone outside both A and B. We can think of the split zone as being both inside and outside C. To describe how C is added at the abstract level, we state which zones are to be inside C and which are to be outside C; those which are split are stated as being both inside and outside. Thus, the inside zones and outside zones between them include all of the zones.

Definition 5 Let $d=(l,Z)$ be an abstract description. Let L be a label in $\mathscr{L}$ and let *in* and *out* be two subsets of Z such that $in \cup out = Z$ and $\emptyset \in out$. Then $d+(L,in,out)$ is defined to be

$$d+(L,in,out)=(l\cup\{L\},Z_{in}\cup Z_{out})$$

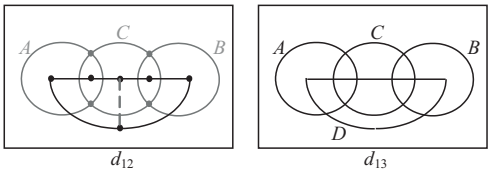
where $Z_{in}=\{z\cup\{L\} : z\in in\}$ and $Z_{out}=out$. If $in=Z$ and $out=Z$ and $L\notin l$ then $d+(L,in,out)$ is said to be derived from d by splitting each zone.

In figure 5, $in=\{\{B\},\emptyset\}$ and $out=\{\{A\},\emptyset\}$; when adding C to d_{10} to give d_{11}, C contains the zone $\{B\}$ (in the set $in-out$), splits the zone $\emptyset$ (in the set $in\cap out$) and does not contain the zone $\{A\}$ (in the set $out-in$). The following lemma shows that the notions of splitting each zone at the abstract and concrete level coincide.

Lemma 2 *Let $d=(Curve,l)$ be a concrete Euler diagram. Let L be a label in $\mathscr{L}$ and let c be a curve that is not in Curve. Then $d+(c,L)$ is derived from d by splitting each zone if and only if $ab(d+(c,L))$ is derived from $ab(d)$ by splitting each zone.*

5 Adding Curves to Wellformed Atomic Layouts

Intuitively, when adding the curve to an atomic diagram, we are seeking a path that cuts zones to be split in two, contains certain zones, excludes the remaining zones, and returns to its starting point. First, we consider the special case of splitting each zone. To illustrate, if we wish to add a curve to d_4 in figure 3 that splits each zone then we can do so by finding a Hamiltonian cycle



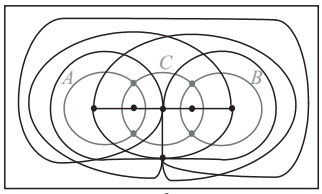
d_{12} d_{13}

Figure 6: Splitting zones and Hamiltonian cycles.

in the concrete dual (figure 4, d_9), as shown in figure 6, d_{12}. This Hamiltonian cycle then, rather nicely, gives us the new curve and wellformedness is maintained, as shown in d_{13}.

Of course, there are concrete dual graphs that have different layouts, but corollary 1 establishes that this is not an issue when splitting each zone. The following theorem, importantly, provides a constructive method for embedding the new curve, namely find a Hamiltonian cycle in an arbitrary concrete dual graph and use that cycle as the image of the new curve.

Theorem 1 *Let $d = (Curve, l)$ be an atomic wellformed concrete Euler diagram containing at least two curves. Let L be a label in $\mathscr{L} - im(l)$. There exists a curve, c, that is not in Curve such that $d + (c, L)$ is wellformed and derived from d by splitting each zone if and only if a concrete dual graph of d is Hamiltonian.*



d_{14}

Figure 7: An extended Euler dual.

There are topologically different ways of adding a curve that splits each zone[1]. In fact, given all concrete dual graphs (reduced by equivalence up to an isotopy of a subset of $\mathbb{R}^2$; we omit the details) we can exactly classify the number of topologically different ways of adding such a curve. To do this, we define a new type of dual graph for atomic diagrams that generalizes both the concrete dual graph and the Euler graph dual: the **extended Euler dual**. Its construction is the same as the Euler graph dual, except that it has additional edges as follows. Given an edge in the Euler graph dual that connects a vertex, v_1, to the vertex, v_2, in the infinite face, we had a choice about the direction that edge wraps around the curves. For each such edge, we add a new edge incident with v_1 and v_2 that wraps the opposite way around the curves. Note that the extended Euler dual is not necessarily planar; in fact, it is only planar when exactly one zone is topologically adjacent to the infinite face. We add the new edges in such a manner

[1] Given d and curves c_1 and c_2 added to d, $d + (c_1, L)$ is topologically different to $d + (c_2, L)$ if c_1 and c_2 are not isotopic in $\mathbb{R}^2 - V(G)$ where $V(G)$ is the set of (images of the) vertices in the Euler graph.

that a minimal number of edge crossings are introduced. The minimal number of crossings is $1+2+...+(\frac{deg(v_2)}{2}-1)$. The construction is illustrated in figure 7, where d_{14} shows the extended Euler dual of d_4 in figure 4.

These extra edges are required in order to make the graph reflect all possible ways (up to some notion of equivalence) of adding a new curve and maintaining wellformedness. We note that we only add extra edges incident with v_2 since the infinite face is the only zone that is not simply connected in an atomic wellformed diagram. In simply connected faces, there is essentially no choice about the 'direction' of the edges.

Theorem 2 *The set of all plane Hamiltonian cycles in the extended Euler dual of an atomic wellformed concrete Euler diagram, d, gives all the topologically different ways of introducing a curve that splits each zone in d and maintains wellformedness.*

The method of adding a curve that splits each zone generalizes. When we want to add a curve that splits a specified set of zones, we instead seek a plane, simple cycle[2] in the extended Euler dual that passes through exactly the vertices corresponding to the zones that are to be split. Suppose that we wish to add a curve to d_4 in figure 3 such that the zone inside exactly C is split as is the zone outside all curves. Then there are several ways of doing this, resulting in diagrams with different abstract descriptions. The method is to find a simple cycle in the extended Euler dual that contains precisely the vertices inside these two zones. One such cycle is shown in d_{15}, figure 8, which results in the curve D being added as shown in d_{16}. Another cycle gives d_{17}, which has an abstraction different from that of d_{16}. Typically, we want to specify which zones are to be contained by the new curve as well as those which are to be split. A method for adding an appropriate curve is captured by the following theorem.

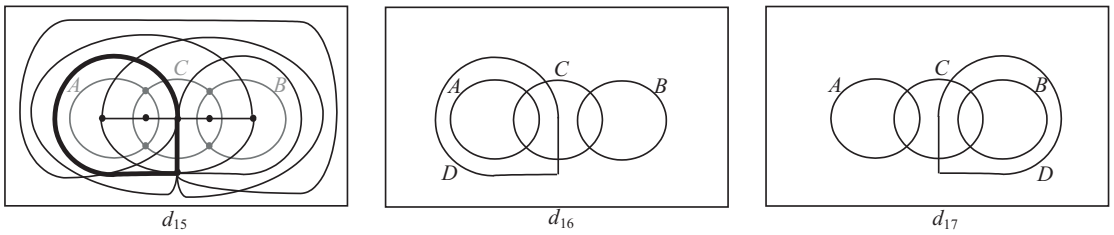


Figure 8: Splitting specified zones.

Theorem 3 *Let $d = (Curve, l)$ be an atomic wellformed concrete Euler diagram. Let L be a label in $\mathscr{L} - im(l)$. Given $ab(d) = (l, Z)$, let $in \subseteq Z$ and $out \subseteq Z$ be such that $in \cup out = Z$ and $\emptyset \in out$. Then there exists a curve, c, that is not in Curve such that $d + (c, L)$ is atomic, wellformed and has abstraction $ab(d) + (L, in, out)$ if and only if either*

1. *there exists a plane, simple cycle, C, in the extended Euler dual, G, such that*

[2] A simple cycle is a cycle, containing at least one vertex, that does not pass through any vertex more than once. A plane cycle is a cycle in which no pair of edges cross.

(a) *the vertices in G that correspond to zones that are elements of the set $in \cap out$ are exactly those in C,*

(b) *the vertices in G that correspond to zones that are elements of the set $in - out$ are located inside C, and*

(c) *the vertices in G that correspond to zones that are elements of the set $out - in$ are located outside C,*

or

2. $|in| = |in \cap out| = 2$ *and the two concrete zones corresponding to the abstract zones that are elements of the set $in \cap out$ are topologically adjacent.*

Again, it is the plane, simple cycle that provides the image of the curve to be added except in case 2, where we do not seek a such a cycle. Instead, we are effectively seeking a path of length 1 (indicating the adjacency of the two zones) and can simply add a curve that is a circle, for example, in the appropriate manner.

6 Adding a Curve to Wellformed Nested Layouts

The previous section characterizes exactly when a curve can be added to an atomic wellformed diagram and maintain wellformedness. Moreover, the characterization provides a constructive method to add a curve to give a new diagram with some specified abstract syntax. Here we demonstrate how to extend the approach to the nested case by example only, due to space limitations. In figure 9, we may want to add a new curve to d_{18} that splits the zone outside all curves, that inside just B and that inside just D, and all remaining zones are outside the curve. To do this, we decompose the diagram into its atomic components, add the new curve, E, to each part in the required manner and then recompose the diagram, joining up the curves labelled E in each of the atomic parts to create a single curve labelled E, shown in d_{19}. The curve E is called a *disconnecting curve* for d_{19}, the theory of which is developed in [FF08].

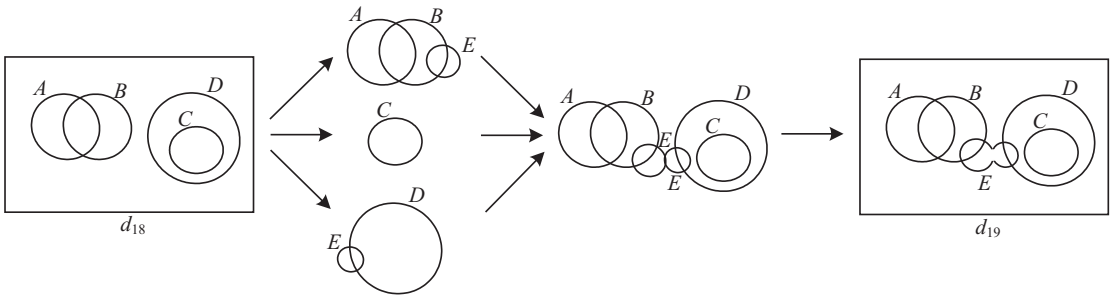


Figure 9: Adding curves to wellformed nested diagrams.

In summary, the approach is to add the required curve to each atomic component and then join the pieces of the new curve together, to create one curve. The results of the previous section tell

us when the curve can be added to the atomic pieces. The only further requirement is that these new curves can be joined together to create an appropriate curve in the original nested diagram. We note that being able to add a curve to each atomic part does not imply that it is possible to add that curve to the entire diagram. For example, suppose we had wanted to add a curve to d_{18} so that instead of splitting the zone inside just D we split the zone inside both C and D. It would not then be possible to join the curves labelled E in the atomic components together to form an appropriate curve in d_{18} whilst maintaining wellformedness. The following theorem characterizes the case when we wish to split each zone.

Theorem 4 *Let d be a wellformed concrete Euler diagram containing exactly n atomic components, say d_1, ... , d_n. Let L be a label in $\mathscr{L} - im(l)$. Then there exists a curve, c, such that $d+(c,L)$ is wellformed and derived from d by splitting each zone if and only if each d_i has a a Hamiltonian concrete dual or contains exactly one curve.*

In the more general wellformed case (where we do not necessarily wish to split every zone), we must be able to add a curve, c, to each atomic component in the required manner (as illustrated above). We need to know when the curves added to the atomic components can be joined up to form c. Suppose that c is to be added to a diagram, d, consisting of two atomic components d_1 and d_2, with d_2 nested in a zone z_1 of d_1. Then the curves c_1 and c_2 added to d_1 and d_2 respectively can be joined whenever they both pass through the zone z_1: in other words, in d_1 the curve c_1 splits z_1 and in d_2 the zone outside all of the curves is split by c_2. This observation generalizes to the case when there are more atomic components.

7 Adding Curves In General

We aim to be able to generate an embedding of any abstract description, which is not possible when imposing the wellformedness conditions. Thus, we no longer insist that the wellformedness conditions are met and allow ourselves to add a curve to arbitrary Euler diagrams. There are many ways of adding curves in the general case and it can be shown that methods exist that allow the inductive construction of a concrete diagram (i.e. by successively adding contours) for any abstract description. Here we informally outline one such method, but better layouts can be achieved by using more sophisticated techniques; for space reasons we do not provide details. Since we are allowed multiple label use, the method may add many curves (which constitute a single contour) in order to achieve the correct zone set after addition.

One point to note is how the interior of a contour is defined when multiple curves have the same label; such a definition is given in [SRH+07] and extends work in [VV04]. To illustrate, in figure 10, d_{20} contains three curves labelled A, which we will refer to as the contour A. A point is interior to the contour A if the number of curves labelled A to which it is interior is odd, otherwise it is exterior to the contour A. Thus, the zones in the diagram are

$$Z(d_{20}) = \{\emptyset, \{A\}, \{A,B\}, \{B\}, \{C\}, \{A,C\}\}$$

and the zone $\emptyset$ is disconnected.

Suppose, to the abstract description $ab(d_{20})$, we wish to add a contour labelled D, given $in = \{\emptyset, \{A\}, \{C\}\}$ and $out = \{\emptyset, \{A\}, \{A,B\}, \{B\}, \{A,C\}$. The diagram d_{21} has the required

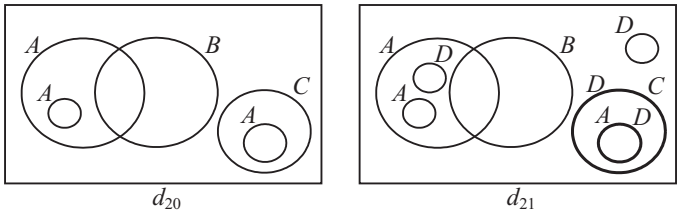


Figure 10: Inside contours and adding curves in general.

abstract description and is obtained from d_{20} by following the following method. Given a concrete diagram $d = (Curve, l)$, a label $L \in \mathscr{L} - im(l)$, and sets $in, out \subseteq Z(ab(d))$ such that $in \cup out = Z(d)$, we can add a curve to d such that:

1. for each abstract zone in the set $in \cap out$, add a curve labelled L properly inside some minimal region of which the corresponding concrete zone consists,

2. for each abstract zone in the set $in - out$, add a curve labelled L for each boundary of any minimal region of which the corresponding concrete zone consists (note that minimal regions may have many boundaries) such that the image of the curve is that boundary, and

3. if $in = \emptyset$ then add a curve labelled L whose image is a straight line segment.

The resulting diagram has abstraction $ab(d) + (L, in, out)$.

8 Application to the Generation Problem

As stated earlier, the generation problem is to find an embedding of any given abstract description (or, at least, each atomic component). Several approaches have been devised to date and they all proceed to find a layout of the entire diagram and do not always succeed. Our results provide a new approach to generation, in that we can inductively produce the required diagram. Suppose we have an abstract description, $d_1 = (l, Z)$, that we wish to draw. In both the wellformed and non-wellformed cases, we can decompose d_1 into a sequence of layout problems, starting with the diagram $(\emptyset, \{\emptyset\})$ and successively adding curves in the required manner to build $d_1 = (l, Z)$. Of course, there are choices about how to inductively add the curves to achieve a good final layout, but there are some obvious methods one can use to help here.

If we want to produce a wellformed layout then we can detect whether the diagram is atomic at the abstract level and consider the abstract components separately, as discussed above. Moreover, there may be some label, l, and abstract description, d_2, such that $d_2 + (l, in, out) = d_1$, for some sets in and out, and d_2 contains more atomic components than d_1; in such cases, it may be more efficient to add the curve labelled l last. To illustrate, we provide an example in figure 11, which produces a wellformed embedding of $d = (l, Z)$ where $l = \{A, B, C, D\}$ and

$$Z = \{\emptyset, \{A\}, \{B\}, \{A,B\}, \{A,C\}, \{A,D\}, \{A,B,C\}, \{A,B,D\}, \{A,C,D\}, \{A,B,C,D\}\}.$$

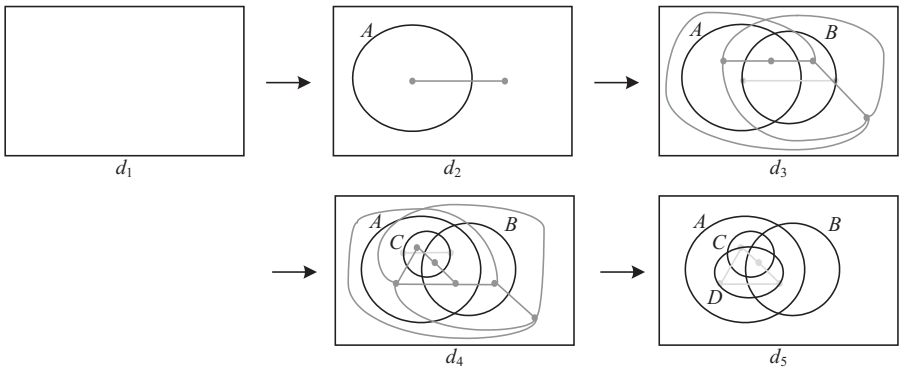


Figure 11: Application to the general generation problem.

There are limits to the inductive construction method. First, we know of wellformed concrete diagrams that cannot be produced using the inductive approach. One such diagram can be seen in figure 12; the removal of any curve results in a diagram that is not wellformed. In this case, a wellformed diagram with the same abstract description can be generated using the inductive method even though this particular layout cannot be achieved. It remains the subject of future work to establish whether any abstract description that has a wellformed embedding can be drawn using our inductive approach that utilizes the extended Euler dual. However, the general method for adding a contour, outlined in section 7, can be shown to yield an embedding of any abstract description using such an inductive construction.

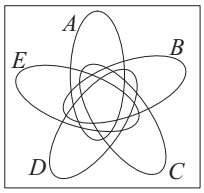


Figure 12: A wellformed Venn-5 embedding.

9 Implementation

To prototype the generation mechanism, we have started implementing the method as a Java program. This draws closed curves with polygons, detects the extended Euler dual using geometric algorithms and routes the edges around this dual as shown in figure 13, where the polygons are regular and form Venn-3. Note that the routing mechanism used for constructing the extended Euler dual means that some edges are very close together and can be mistaken as tangential.

In figure 13, we show the result of finding a Hamiltonian cycle in the extended Euler dual, and using that cycle to add a new polygon, labelled ‘d’ to the diagram, resulting in Venn-4. In this case we chose the new curve so that it intersects with every zone, but we could have

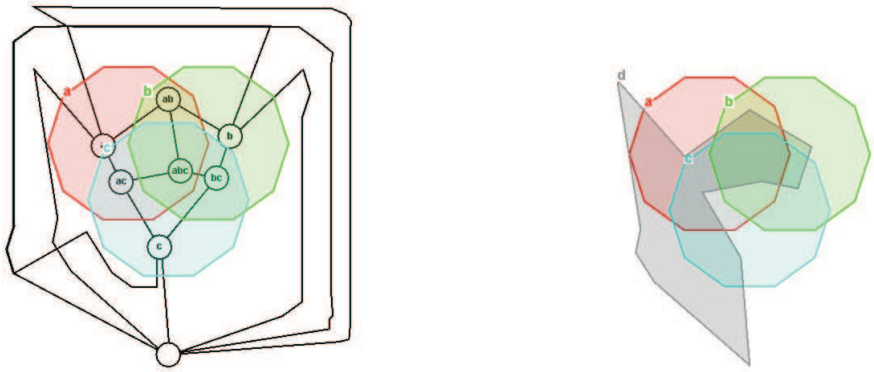


Figure 13: Using the extended Euler dual graph to add a curve.

used any simple cycle in the extended Euler dual to add a curve and maintained wellformedness (of course, the chosen cycle impacts the abstract description). We can enumerate every simple cycle by finding sets of faces in the extended Euler dual that are connected; the cycle formed by traveling around such a set of faces then gives rise to a new curve. Layout improvements, such as those applied in [FRM03], are required in order to improve the appearance of the diagram; this is currently being implemented.

Our intention is to use this generation mechanism to enumerate through possible diagrams, looking for those that can be drawn 'nicely', for instance where a regular polygon can be added to a diagram already consisting of regular polygons. The process of generating the extended Euler dual and discovering a single simple cycle within it is reasonable efficient and works in real time. However, the time complexity of enumerating every cycle is exponential relative to the number of edges and so will be infeasible as the size of the diagrams increases beyond the small diagrams shown in this paper; heuristics will need to be developed for this task.

10 Conclusion

In this paper we have presented several methods for generating Euler diagrams by modifying existing layouts. The technique we have presented to add a curve in the wellformed case guarantees to preserve wellformedness. Moreover this novel technique of using inductive generation methods can be used to produce embeddings of a class of abstract description. Indeed, our general method of adding a curve in the non-wellformed case can be used to generate an embedding of any abstract description.

We plan to use these inductive embedding methods (concentrating initially on wellformed diagrams) to populate a library of drawn examples from which we can subsequently create further embeddings by adding further curves. We anticipate that such a library will contain a concrete diagram for each abstract description with up to three labels, and many with four labels (there are 2^{16} abstract descriptions with four labels). We will then be able to take abstract descriptions and select sub-diagrams from the library and add curves to them to produce the required concrete diagram.

Further work also includes extending the generation algorithms to allow subsets of the well-formedness conditions to be imposed. We anticipate using a hybrid of the Euler graph and the extended Euler dual to allow, for example, concurrency or triple points to be present in the created layouts. This will enable a wider variety of layouts to be produced and will allow us to take user preference more fully into account, for example.

Acknowledgements: This work is supported by the UK EPSRC grants EP/E011160/1 and EP/E010393/1 for the Visualization with Euler Diagrams project. Thanks also to John Taylor for helpful discussions on aspects of this research.

Bibliography

[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. *Proc. Euler Diagrams 2004* Elsevier, ENTCS vol. 134, pages 19–31, 2005.

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

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

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

[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.

[FRM03] J. Flower, P. Rodgers, P. Mutton. Layout metrics for Euler Diagrams. *7th International Conference on Information Visualisation* IEEE Computer Society Press, pages 272-280, 2003.

[FF08] A. Fish, J. Flower. Euler Diagram Decomposition. *accepted for Diagrams 2008*, Springer, 2008.

[HES+05] P. Hayes, T. Eskridge, R. Saavedra, T. Reichherzer, M. Mehrotra, D. Bobrovnikoff. Collaborative Knowledge Capture in Ontologies. *Proc. 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.

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

[KMG+05] 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.

[MEL+95] K. Misue, P. Eades, W. Lai, K. Sugiyama. Layout Adjustment and the Mental Map, *Journal of Visual Languages and Computing*, 2(6):183-210, 1995.

[Nie2006] L. Niebrój. Defining Health/Illness: Societal and/or Clinical Medicine? *Journal of Physiology and Pharmacology*, 57(4):251-262, 2006.

[RZF08] P. Rodgers, L. Zhang, A. Fish. General Euler Diagram Generation, accepted for Diagrams 2008, Springer, 2008.

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

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

[SFR07] G. Stapleton, A. Fish and P. Rodgers. Abstract Euler Diagram Isomorphism. *accepted for Visual Languages and Computing*, Knowledge Systems Institute, 2008.

[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.

[SRH+07] G. Stapleton, P. Rodgers, J. Howse and J. Taylor. Properties of Euler Diagrams. *Proc. of Layout of Software Engineering Diagrams*, EASST, pages 2–16, 2007.

[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.

[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.

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