Changing Euler Diagram Properties by Edge Transformation of Euler Dual Graphs

John Howse
University of Brighton
john.howse@brighton.ac.uk

Peter Rodgers
University of Kent
p.j.rodgers@kent.ac.uk

Gem Stapleton
University of Brighton
g.e.stapleton@brighton.ac.uk

Abstract

Euler diagrams form the basis of several visual modelling notations, including statecharts and constraint diagrams. Recently, various techniques for automated Euler diagram drawing have been proposed, contributing to the Euler diagram generation problem: given an abstract description, draw an Euler diagram with that description and which possesses certain properties. A common generation method is to find a dual graph from which an Euler diagram is subsequently created. In this paper we define transformations of the dual graph that allow us to alter the properties that the generated diagram possesses. In addition, because the dual graph of a previously generated diagram can be found, our transformations can be used to take such a diagram and produce a new diagram with the same abstract description, but with different properties. As a result, we can produce a variety of different diagrams for any given abstract description, allowing us to choose an Euler diagram that conforms to the properties that a user prefers.

1 Introduction

Many diagrams are based on finite collections of closed curves; such a collection of closed curves is called an Euler diagram [3], of which Venn diagrams are examples. Euler diagrams form the basis of several visual modelling notations, including statecharts [5] and constraint diagrams [7]. Euler diagrams also have wide-ranging uses in the area of information visualization, such as [2, 6, 8, 9].

Various methods for automatically generating Euler diagrams have been developed, each concentrating on a particular class of diagrams; for example [1, 4, 8, 13]. Ideally, generation algorithms will produce diagrams with effective layouts. The algorithms developed so far produce Euler diagrams that have certain sets of properties, sometimes called wellformedness conditions. Such wellformedness conditions are chosen since they correlate with usability. We consider five properties: no pair of curves running concurrently, connected zones, no n-points, all of the curves being simple, and the labels of the curves being unique (each property will be explained below). This paper discusses ways of changing them during the generation process.

Each generation method starts with an abstract description of the required diagram and proceeds to seek a layout. It has been shown that not all abstract descriptions can be realized as Euler diagrams with arbitrary sets of the properties described in this paper [10]. This means that some abstractions have to be embedded whilst violating certain properties. Hence, we cannot obtain a generation method that both guarantees to be able to embed any abstract description (that presented in this paper can embed any abstract description) and ensure that all five properties hold. We provide techniques that allow us to alter the properties of the embedded Euler diagram, allowing us to reduce the number of times we violate the wellformedness conditions.

The generation approach that is used in this paper extends those in [4, 11]. Firstly, it constructs a vertex-labelled graph, G, from the abstract description, which is often called the Euler dual of the (to be) generated Euler diagram, d. This graph G is embedded in the plane, and d's closed curves are formed using a dual of G. We use graph transformations on the Euler dual G to change the properties possessed by the generated Euler diagram, focusing on those that either add or delete edges. Section 2 gives an overview of Euler diagrams and their abstraction descriptions. Section 3 describes our process of transforming an abstract description into an Euler diagram, via a vertex-labelled graph, which generalizes other approaches found in the literature. The five properties considered in the paper are introduced in section 4. Sections 5.1 to 5.5 describe the graph transformations that allow us to alter the properties.

2 Euler Diagrams

Figure 1 shows an Euler diagram with three curves labelled a, b and c. It asserts that c is a subset of $b - a$.

An **Euler diagram** $d = (Curve, l)$ is a pair where $Curve$ is a collection of closed curves, and l is a function

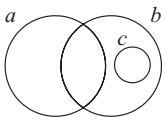


Figure 1. An Euler diagram.

that returns the label of each curve in $Curve$. A closed curve is a continuous function of the form $f\colon [a,b] \to \mathbb{R}^2$ where $[a,b]$ is some interval of real numbers and $f(a) = f(b)$. The labels are taken to be from some fixed set of labels, $\mathcal{L}$ (so, the codomain of l is $\mathcal{L}$).

We define a **contour** with label λ of a diagram, d, to be the set of curves in d with label λ. A **zone** in d is a region of d that lies within some set of contours but outside the remaining contours. The diagram in Figure 1 has five zones, including that inside both a and b but outside c.

In order to generate Euler diagrams, we need a description of the diagram. We can provide an abstract description of an Euler diagram by considering its zones: a zone can be represented by the labels of the contours that contain it.

Definition 1 Elements of $\mathcal{Z} = \mathbb{P}\mathcal{L}$ are called **abstract zones** (or, simply, zones). An **abstract description**, D, is a set of abstract zones, $D \subseteq \mathcal{Z}$ such that $\emptyset \in D$.

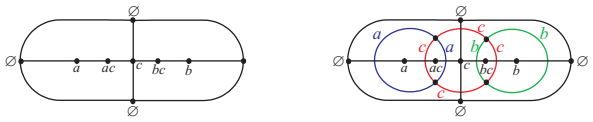
Definition 2 Given an Euler diagram $d = (Curve, l)$, we map d to $ab(d) = D$, called the **abstract description** of d, where D contains exactly one abstract zone for each zone in d; in particular, given a zone, z, in d, the set D contains the abstract zone $ab(z) = \{l(c) : c \in C(z)\}$ where $C(z)$ is the set of curves in d that contain z.

The diagram, d, in Figure 1, has abstraction $ab(d) = \{\emptyset, \{a\}, \{b\}, \{a,b\}, \{b,c\}\}$. For ease of readability, we will abuse notation and write $ab(d) = \{\emptyset, a, b, ab, bc\}$.

3 Graph Based Euler Diagram Generation

The *generation problem* is as follows: given an abstract description, find an Euler diagram with that abstract description. A common method first converts the abstract description into a vertex-labelled graph. An Euler diagram can then be formed, essentially by finding a dual of the graph; recall that a dual graph is formed by placing a vertex in each face of the primal (original) graph and adding edges between vertices when they are in adjacent faces, with each original edge being crossed by a unique dual edge.

Consider the abstract description $D = \{\emptyset, a, ac, c, bc, b\}$. A vertex-labelled graph for D is shown in Figure 2(a). The vertices are labelled with zones from D and the edges are labelled with the symmetric difference of the labels of incident vertices (two vertices are incident if they are joined by an edge). A dual is generated, but ignoring the outer face. The labels of the edges of the original graph inform the labelling of the edges of its dual, shown in Figure 2(b).

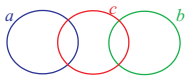


(a) A vertex-labelled graph, G.

(b) G with its dual.

Figure 2. Graph based generation.

Throwing away the original graph and the vertices of the dual, we obtain an Euler diagram, d, with abstraction D:



The labelling of the edges in the dual gives the curves in d that run along that edge. For instance, the two edges labelled a give rise to a closed curve labelled a; similarly for b and c (in c's case, there are four edges). So, the edge-labelled dual graph is essentially the Euler diagram.

Figure 3 shows two distinct duals of the underlying graph G (shown in grey). For example, the vertex in G labelled a is contained by both edges in the lefthand dual graph but contained by only one edge in the righthand dual graph. This is an important observation since we use a dual, G^*, of a graph, G, to embed an Euler diagram. If that Euler diagram is to have the required abstraction, the vertices of G have to be enclosed by the correct curves (further examples will be given below). Not every dual of an arbitrary plane embedding of G necessarily gives rise to an Euler diagram with the required abstraction. Our generation method must account for this, further discussed below.

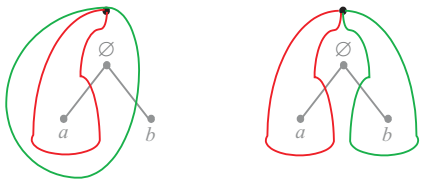


Figure 3. Two distinct dual graphs.

The generation process is outlined in Fig. 4. We start with an abstract description, D, that we want to embed. We then turn D into a vertex-labelled graph, G, such that every abstract zone in D appears as a vertex label in G *and* every vertex label is an abstract zone in D. Various approaches to this stage of the generation process have essentially been considered elsewhere (albeit not in the most general form

we have presented in this paper), such as [4, 11]. We proceed to embed G, giving $\hat{G}$, in such a manner that G has only edges labelled $\emptyset$ next to the infinite face; if this is not possible then we can add edges to G so that it becomes possible (note that we never need to add vertices, but we can choose to if we wish). Having every edge next to the infinite face being labelled $\emptyset$ ensures that $\hat{G}^*$ will give rise to a diagram with abstraction D. We now proceed to define the graphs we need for this generation method.

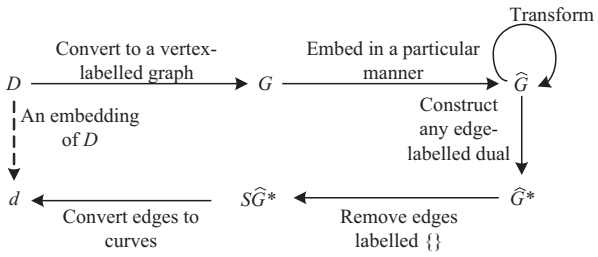


Figure 4. The generation process

Definition 3 A **vertex-labelled graph**, $G = (V, E, l_V, l_E)$, is a graph such that

1. (V, E) is a graph with vertex set V and edge set E,
2. each vertex, v, in V is labelled by a zone, that is $l_V : V \to \mathcal{Z}$, and
3. the vertex labelling induces an edge labelling, $l_E : E \to \mathcal{Z}$, defined by

$$l_E(e) = (l_V(v_1) - l_V(v_2)) \cup (l_V(v_2) - l_V(v_1))$$

where v_1 and v_2 are the vertices incident with e.

We will be constructing edge-labelled dual graphs of vertex labelled graphs. For the purposes of this paper, we only require an informal definition of an edge labelled graph: an **edge labelled graph** is a graph whose edges are labelled by sets of curve labels. That is, we are labelling edges with a set of labels from $\mathbb{P}\mathcal{L}$. The curve labels that are in an edge label tell us exactly which curves in the to-be-generated Euler diagram will run along that edge. For each curve label $\lambda \in \mathcal{L}$ we say that λ is a label of edge e if the curve label λ is in the set of curve labels labelling e, denoted by $l_E(e)$.

Since we want to create a dual graph, we need a drawing of the vertex-labelled graph. A drawing of a graph in $\mathbb{R}^2$ in which no edges cross is called a *plane embedding*. There are various different kinds of dual graph but, since we are creating a dual from a plane embedding (as described above), we produce a *geometric dual*.

Definition 4 *Let $G = (V, E, l_V, l_E)$ be a planar, vertex-labelled graph with a plane embedding $\hat{G}$. Let $\hat{G}^* = (V^*, E^*, l_{E^*})$ be an edge-labelled graph that is a geometric dual of $\hat{G}$. If for each edge, $e^* \in E^*$, $l_{E^*}(e^*) = l_E(e)$ where e^* crosses e then the (plane, embedded) graph $\hat{G}^*$ is called an **edge-labelled dual** of $\hat{G}$.*

The main focus of this paper is on the transformations that can be applied to $\hat{G}$ in order to change the properties possessed by the embedded Euler diagram. After applying transformations to $\hat{G}$, an edge-labelled dual, $\hat{G}^*$, of $\hat{G}$ is constructed. The labels on the edges of $\hat{G}^*$ indicate that curves with those labels run along that edge. Hence, edges labelled $\emptyset$ will not be traversed by any curves and are deleted, to give a graph $S\hat{G}^*$. At this point, $S\hat{G}^*$ has the same image as the Euler diagram we will create. The final step is to turn $S\hat{G}^*$ into an Euler diagram (i.e. a set of closed curves together with a labelling function). The method we develop to produce a set of curves ensures that they have the correct containment properties. That is, a vertex labelled z is embedded in a zone with abstraction z. To create the curves, we need access to certain subgraphs of $\hat{G}^*$.

Definition 5 *Let $G = (V, E, l_E)$ be an edge-labelled graph and let λ be a curve label in $\mathcal{L}$. The subgraph of G obtained by deleting all edges whose labels do not contain λ (along with any isolated vertices), denoted $G_E^+(\lambda)$, is the **in-edge subgraph** of G given λ. The subgraph of G obtained by deleting all edges whose labels contain λ (along with any isolated vertices), denoted $G_E^-(\lambda)$, is the **out-edge subgraph** of G given λ.*

Each subgraph $G_E^{*+}(\lambda)$ is essentially the contour λ in the generated Euler diagram. We note that every $G_E^{*+}(\lambda)$ has only Eulerian components and the curves labelled λ are formed by traversing Eulerian cycles in these components. As a simple illustration of the process of converting an Eulerian cycle to a curve, consider the graph, $\hat{G}$, in Fig. 5, together with its dual $\hat{G}^*$. The graph $\hat{G}_E^{*+}(a)$ is shown and a curve labelled a can be formed by, intuitively, traversing the edges of $\hat{G}_E^{*+}(a)$, following an Eulerian cycle.

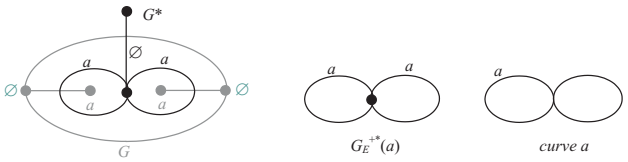


Figure 5. Converting cycles to curves

Given an embedding, $\hat{G}$, of a connected, planar, vertex-labelled graph the Euler diagram, d, obtained by traversing the components of each $G_E^{*+}(\lambda)$ in this manner is said to be **generated** from $\hat{G}$; the graph $\hat{G}$ is the **Euler dual** of d. The Euler diagram has the correct abstraction provided $\hat{G}$ possesses certain properties.

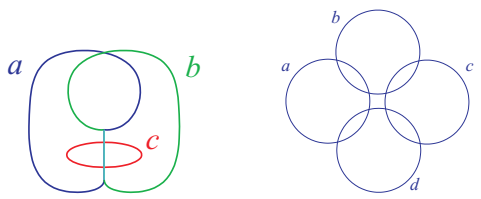
Definition 6 *Let $G = (V, E, l_V, l_E)$ be a planar, vertex-labelled graph with embedding $\hat{G}$. If $\hat{G}$ is plane, every edge, e, embedded next to the infinite face of $\hat{G}$ has label $l_E(e) = \emptyset$ and is incident with a vertex, v, whose label is $l_V(v) = \emptyset$, then $\hat{G}$ is called an **appropriate** embedding of G.*

Theorem 1 *Let $G = (V, E, l_V, l_E)$ be a connected, planar, vertex-labelled graph with an appropriate embedding, $\hat{G}$. Let $d = (Curve, l)$ be an Euler diagram generated from $\hat{G}$. Then $ab(d) = im(l_V)$.*

4 Euler Diagram Properties

An Euler diagram may possess the following properties [12]:

1. **No concurrency** No pair of curves run concurrently.
2. **Simplicity** All of the curves are simple (i.e no curve self-intersects).
3. **No n-points** There are no n-points of intersection between the curves (i.e. there are no points that the curves pass through at least n times).
4. **Connected zones** Each zone is a connected component of $\mathbb{R}^2$; a connected component is a minimal region.
5. **Unique labels** Each label is used on at most one curve.



(a) Concurrency and a 3-point.

(b) A disconnected zone.

Figure 6. Properties.

In Figure 6(a), the diagram has a and b running (partially) concurrently, and triple points where c intersects with both a and b. In Figure 6(b), the diagram has a disconnected zone: that outside all four curves has two components.

For each of these five properties, we can count the number of times a diagram 'violates' them. Whilst it is desirable that a diagram possess all properties, counting them allows us to identify when we have reduced their occurrence; we reduce the number of violations in embedded diagrams using graph transformations. For non-simplicity, we can count the number of times a curve self-intersects: given a closed curve c: $[0, 1] \to \mathbb{R}^2$, c self-intersects $|\{a \in [0, 1) : \exists b \in [0, 1) - \{a\}\, c(a) = c(b)\}|$ times. Generalizing this to contours, the number of times a contour self-intersects is the sum of the number of times its curves self-intersect, plus the number of pairwise intersections between its curves.

For concurrency, we can count the number of concurrent line segments as a measure of how much concurrency is present. For instance, in Figure 6(a), there are three concurrent line segments: that inside c that consists of a and b, and the two outside c that also consist of a and b.

In the case of disconnected zones, we can count the number of minimal regions and subtract the number of zones. The case for non-unique labels is equally simple: count the number of curves and subtract the number of labels. Finally, for n-points, we can count the number of n-points in the diagram. In this paper, we define graph transformations that allow us to reduce the count of these properties, thereby obtaining a 'more wellformed' Euler diagram.

5 Dual Graph Transformations

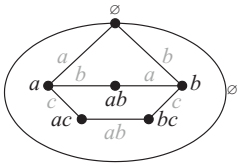
The Euler dual, $G = (V, E, l_V, l_E)$, from which we generate an Euler diagram, d, determines the properties that d will possess. In order to change these properties, we can transform G. Let e be an edge whose incident vertices are in G and define

1. $G + e$ to be the graph $(V, E \cup \{e\}, l_V, l_E \cup \{(e, z)\})$ where z is the symmetric difference of the label sets of the vertices incident with e, and
2. $G - e$ to be the graph $(V, E - \{e\}, l_V, l_E - \{(e, l_E(e))\})$.

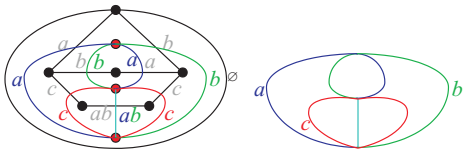
Note that we will be applying the above transformations to embedded graphs. When deleting an edge the embedding of the remaining components remains unchanged. Similarly, when adding an edge the embedding of the pre-existing vertices and edges remains unchanged; we will specify information as to how the new edge is to be embedded (such as across a specified face).

5.1 Concurrency

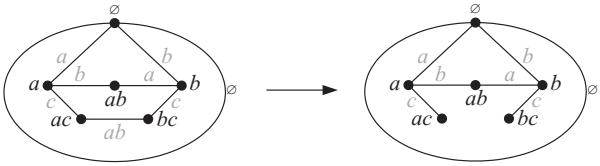
Two curves will run concurrently in a generated Euler diagram whenever the Euler dual, G, contains an edge that is labelled with more than one curve label. This, of course, is independent of any embedding of G. Consider the following graph for the abstract description $D = \{\emptyset, a, b, ab, ac, bc\}$:



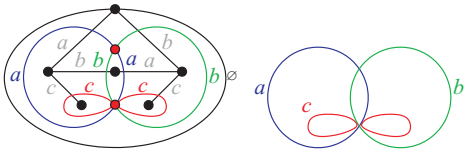
This graph has an edge labelled ab and thus curves a and b will run concurrently in the Euler diagram it generates:



To remove concurrency from a diagram, we simply delete multiply-labelled edges. Deleting the edge labelled ab from the vertex-labelled graph we obtain:



generating an Euler diagram without concurrency:



The resulting Euler diagram has a non-simple curve; however, deleting a multiply-labelled edge will not always result in the generation of a diagram with non-simple curves. In the next section, we consider how to reduce the number of self-intersections, resulting in simple curves.

Definition 7 *Let $G = (V, E, l_V, l_E)$ be a vertex-labelled graph. Let e be an edge in G such that $|l_E(e)| \geq 2$. Then e is called a **concurrency edge** in G.*

The number of concurrency edges in a diagram is precisely the number of concurrent line segments in the generated Euler diagram.

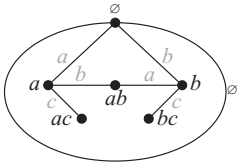
Theorem 2 *Let $G = (V, E, l_V, l_E)$ be a connected, planar vertex-labelled graph with an appropriate embedding $\hat{G}$. Let e be a concurrency edge in G such that $G - e$ is connected. Then*

1. *$\hat{G} - e$ is an appropriate embedding of $G - e$,*
2. *the Euler diagram, d_1, generated from $\hat{G}$ has the same abstraction as d_2 generated from $\hat{G} - e$, and*
3. *d_2 has less concurrency than d_1.*

We note that it is not necessarily possible to remove all concurrency from within an Euler diagram, such as from those with abstraction $\{\emptyset, ab\}$. However, we can use this transformation to reduce the concurrency in a diagram, up to some limit: we can continue to remove concurrency edges until removing any such edge disconnects the graph. We could, for instance, choose to prioritize deleting concurrency edges that have more labels.

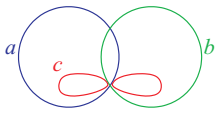
5.2 Simplicity

In order to detect non-simple curves in a generated Euler diagram, we consider a particular embedding of the vertex-labelled graph. An *edge sequence* of a face is a sequence of labels of the edges forming a minimal closed walk that includes all edges around that face. Edge sequences for the two internal faces of the vertex-labelled graph

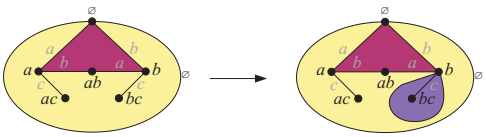


are a, b, a, b and, starting with the righthand edge labelled a and then traversing the dangling edge labelled c, $a, c, c, b, \emptyset, a, c, c, b$; in the former case, a, b, a, b, a, b, a, b is not an edge sequence for the same face since it is not minimal (its length is longer than a, b, a, b). Recall that, in general, edge labels are sets of curve labels.

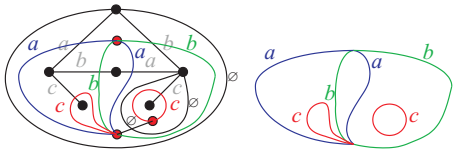
A generated diagram will contain a non-simple curve if and only if there is a face in which a curve label occurs more than twice in some edge sequence. The label c occurs four times in the sequence $a, c, c, b, \emptyset, a, c, c, b$ for the graph above and the generated diagram does indeed include a non-simple curve c:



To reduce the number of self-intersection points we must, therefore, add edges to the vertex-labelled graph so that no label occurs more than twice in an edge sequence of any face. In our running example there are several ways of doing this. We could add an edge between the vertices labelled ac and bc; this would reverse the process described in the previous section and result in a concurrent edge. Alternatively, we could perform the following transformation:



We have added a loop around a dangling edge. This results in the creation of a new face (the faces are shaded in each graph) and now no curve label occurs more than twice in any edge sequence of any face. The generated diagram contains no non-simple curves:



The resulting diagram contains curves with non-unique labels: two curves are labelled c.

Given an edge sequence $ES = L_1, L_2, ..., L_n$ we can associate with that sequence a vertex sequence, $V = v_1, ..., v_{n+1}$ where v_i and v_{i+1} are the vertices incident with the edge, e_i that gave rise to L_i, reflecting the order in which we walk along e_i to give ES. We note that any curve label λ occurs an even number of times in an edge sequence.

Definition 8 *Let $G = (V, E, l_V, l_E)$ be a connected, planar, vertex-labelled graph with an appropriate embedding $\hat{G}$ which, in turn, has a face, f, with an edge sequence $ES = L_1, L_2, ..., L_n$ in which a curve label, λ, occurs more than twice. Let $ES_1 = L_1, ..., L_j$ and $ES_2 = L_{j+1}, ..., L_n$ be a subdivision of ES such that λ occurs at least twice in both ES_1 and ES_2. Let e be an edge that is not in $\hat{G}$ whose incident vertices are v_1 and v_{j+1}. Then e is called a* ***self-intersection reducing edge*** *for λ and $\hat{G}$ in f.*

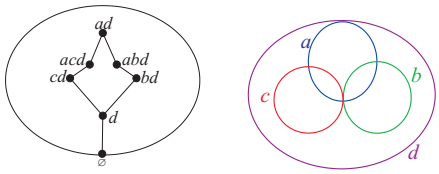
Theorem 3 *Let $G = (V, E, l_V, l_E)$ be a connected, planar vertex-labelled graph with an appropriate embedding $\hat{G}$. Let e be a self-intersection reducing edge for λ and $\hat{G}$ in f. Add e to $\hat{G}$, giving $\hat{G} + e$, by embedding e across f, ensuring planarity is maintained. Then*

1. *$\hat{G} + e$ is an appropriate embedding of $G + e$,*
2. *the Euler diagram, d_1 generated from $\hat{G}$ has the same abstraction as d_2 generated from $\hat{G} + e$,*
3. *the contour labelled λ in d_2 has fewer self-intersection points than in that labelled λ in d_1, and*
4. *every other contour in d_2 has the same number or fewer self-intersection points than that with the same label in d_1.*

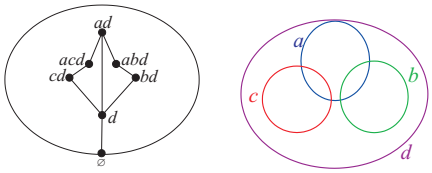
One can always apply this type of transformation to $\hat{G}$. Whenever we have a non-simple curve, we can find an edge to add to $\hat{G}$ to reduce the number of self-intersections that the curve (strictly, contour) possesses. In the limit, this allows us to remove all self-intersections from the embedded Euler diagram and, thus, use only simple curves.

5.3 n-points

Given an embedding of a vertex-labelled graph for an abstract description, an n-point arises if the edge sequence around a face contains at least $2n$ occurrences of labels.The abstract description $\{\emptyset, abd, acd, ad, bd, cd, d\}$ has vertex-labelled graph, together with a generated Euler diagram:



This diagram contains a triple-point (a 3-point). In order to separate the curves, we can add an edge across a face, as shown below with the new generated diagram:



In the following definition, $sd(v_i, v_j)$ denotes the symmetric difference of the label sets of vertices v_i and v_j.

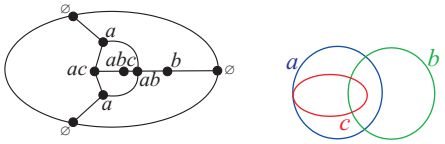
Definition 9 *Let $G = (V, E, l_V, l_E)$ be a connected, planar, vertex-labelled graph with an appropriate embedding $\hat{G}$ which, in turn, has a face, f, with an edge sequence $ES = L_1, L_2, ..., L_m$ where $\sum_{1 \leq i \leq m} |L_i| \geq n$ for some n. Suppose that $ES_1 = L_1, ..., L_j$ and $ES_2 = L_{j+1}, ..., L_n$ is a subdivision of ES such that $|sd(v_1, v_{j+1})| + \sum_{1 \leq i \leq j} |L_i| < n$ and $|sd(v_1, v_{j+1})| + \sum_{j+1 \leq i \leq m} |L_i| < n$. Let e be an edge that is not in $\hat{G}$ whose incident vertices are v_1 and v_{j+1}. Then e is called an* ***n-point reducing edge*** *for $\hat{G}$ in f.*

Theorem 4 *Let $G = (V, E, l_V, l_E)$ be a connected, planar vertex-labelled graph with an appropriate embedding $\hat{G}$. Let e be an n-point reducing edge in $\hat{G}$ in some face f. Add e to $\hat{G}$, giving $\hat{G}+e$, by embedding e across f, ensuring the plane embedding is maintained. Then*

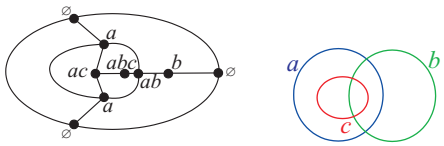
1. *$\hat{G} + e$ is an appropriate embedding of $G + e$,*
2. *the Euler diagram, d_1 generated from $\hat{G}$ has the same abstraction as d_2 generated from $\hat{G} + e$, and*
3. *d_2 has (exactly one) fewer n-points than d_1.*

5.4 Connected zones

A zone z in a generated Euler diagram is connected if and only if the vertices labelled z in the generating vertex-labelled graph are connected. Consider the abstract description $D = \{\emptyset, a, ab, abc, ac, b\}$. A vertex-labelled graph for D, together with the generated Euler diagram, is:



The zone that is inside a is disconnected. We can transform the graph, adding an edge to 'reconnect' this zone:



Definition 10 *Let $G = (V, E, l_V, l_E)$ be a connected, planar, vertex-labelled graph with an appropriate embedding $\hat{G}$ which, in turn, has a face, f, that is bounded by edges that are incident with two distinct vertices, v_1 and v_2 that are not adjacent and have the same non-empty label, that is $l_V(v_1) = l_V(v_2) \neq \emptyset$. Let e be an edge that is not in $\hat{G}$ whose incident vertices are v_1 and v_2. Then e is called a* ***disconnected zone reducing edge*** *for f in $\hat{G}$.*

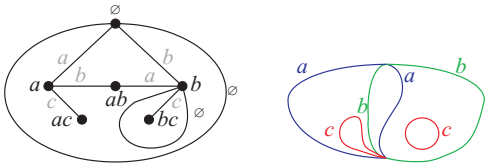
Theorem 5 *Let $G = (V, E, l_V, l_E)$ be a connected, planar vertex-labelled graph with an appropriate embedding $\hat{G}$. Let e be a disconnected zone reducing edge for f in $\hat{G}$. Add e to $\hat{G}$, giving $\hat{G}+e$, by embedding e across f, ensuring the plane embedding is maintained. Then*

1. *$\hat{G} + e$ is an appropriate embedding of $G + e$,*
2. *the Euler diagram, d_1 generated from $\hat{G}$ has the same abstraction as d_2 generated from $\hat{G} + e$, and*
3. *d_2 has fewer minimal regions than d_1.*

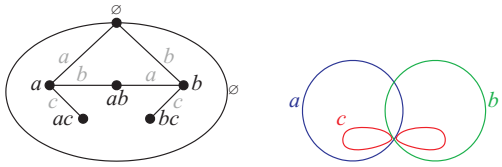
In particular, the zone with abstraction $l_V(v_1)$ consists of one fewer minimal regions in d_2 than in d_1.

5.5 Non-unique curve labels

A single curve will be generated for a given label λ from an embedding of the vertex-labelled graph if all edges labelled λ are contained in a face of the graph $\hat{G}^-_E(\lambda)$. This is not the case in the vertex-labelled graph immediately below, where edges labelled c are located in two different faces of $\hat{G}^-_E(\lambda)$; the generated diagram is shown on the right.



To reduce the number of non-unique labels in the generated Euler diagram, we delete edges from the vertex-labelled graph to reduce the number of faces in $\hat{G}^-_E(\lambda)$ that contain edges of $\hat{G}$ labelled λ. In this example, we would remove the loop labelled $\emptyset$ from around one of the edges labelled c, so that the two edges labelled c lie in the same underlying face to produce:



In this case, it can be beneficial to delete many edges to reduce the number of non-unique labels. We use the notation $G - E$ to mean $G - e_1 - e_2 ... - e_n$ where $E = \{e_1, e_2, ..., e_n\}$.

Definition 11 *Let $G = (V, E, l_V, l_E)$ be a connected, planar, vertex-labelled graph with an appropriate embedding $\hat{G}$. Let E' be a set of edges in $\hat{G}^-_E(\lambda)$ for some curve label λ. If*

1. *$\hat{G}^-_E(\lambda) - E'$ is connected,*
2. *no edge in E' bounds the infinite face of $\hat{G}$,*
3. *the number of faces in $\hat{G}^-_E(\lambda)$ that contain edges of $\hat{G}$ labelled λ is greater than the number of faces in $\hat{G}^-_E(\lambda) - E'$, and*
4. *for all subsets of E' the number of faces in $\hat{G}^-_E(\lambda)$ that contain edges of $\hat{G}$ labelled λ is not greater than the number of faces in $\hat{G}^-_E(\lambda) - E'$ (i.e. E' is minimal)*

then E' is a called an ***non-unique label reducing set of edges*** *in $\hat{G}$ for λ.*

Theorem 6 *Let $G = (V, E, l_V, l_E)$ be a connected, planar vertex-labelled graph with an appropriate embedding $\hat{G}$. Let E' be a set of non-unique label reducing edges in $\hat{G}$ for some λ. Then*

1. *$\hat{G} - E'$ is an appropriate embedding of $G - E'$,*
2. *the Euler diagram, d_1 generated from $\hat{G}$ has the same abstraction as d_2 generated from $\hat{G} - E'$,*
3. *d_2 has fewer non-unique labels than d_1, that is $|Curve_2| - |im(l_2)| < |Curve_1| - |im(l_1)|$ where $Curve_i$ and l_i denote the set of curves and the labelling function respectively for d_i.*

5.6 Interrelationships Between Transformations

When we apply one of the above transformations to reduce the number of violations of a particular property, it may well be that we are increasing the number of violations of another property. For example, if we remove a concurrency edge then we may increase the number of curve self-intersections in a diagram. This is because removing an edge results in two adjacent faces of the graph being

merged in to one; this can be seen in the figures given in the concurrency section, where the edge, e, removed from the original graph, G, is a self-intersection reducing edge for $G - e$ (that is, adding e to $G - e$ reduces the number of self-intersections).

Adding a self-intersection reducing edge to a graph G can often have the effect of increasing the number of curves of which a contour consists, thereby giving rise to more non-unique labelling. When reducing the number of self-intersections, we can prioritize adding edges that introduce a minimal amount of concurrency (i.e. choose to add edges whose edge labels contain fewer curve labels). Moreover, we can choose to add an edge which resolves non-simple points for many curves, rather than add many edges to resolve the respective self-intersection points. When applying other transformations, we note that adding an edge never increases non-simplicity.

Applying the other transformations described in this paper can also have negative consequences for the number of violations of other properties. For example, adding a disconnected zone reducing edge could lead to the need for more non-unique labelling. We need to find a careful balance between reducing the number of violations of one property at the expense of another. The choice about the 'best' edges to add or remove can be informed by user preference.

6 Conclusion

Euler diagrams form the basis of many notations associated with visual modelling and are often used for information visualization. Enabling the automated generation of effective Euler diagram layouts will be of aid in numerous areas. We have defined a series of graph transformations on the Euler dual that enable the properties of the generated Euler diagrams to be altered prior to their generation. Furthermore, we can use the results presented in this paper to generate alternative layouts of previously embedded Euler diagrams: generate a dual graph of the Euler diagram and proceed to apply transformations to that graph in order to change the layout.

The transformations described in this paper form the basis of several avenues of future research. In particular, we plan to develop sophisticated searches through the space of vertex-labelled graphs, using the transformations, enabling us to effectively choose a graph from which to generate the desired Euler diagram. The choice of graph has a profound impact on the resulting embedded Euler diagram and, therefore, needs to be selected with care.

Second, we considered only edge transformations. Adding or removing vertices can also have a profound effect on the appearance of the embedded diagram, again altering the properties possessed. The next stage in this work is to consider combinations of vertex and edge transformations and their impact on the embedded Euler diagrams. Finally, we also plan to investigate how to choose an effective embedding of the chosen vertex-labelled graph prior to applying transformations. The different embeddings of a graph also greatly impact the appearance of the generated Euler diagram. Once we have a mechanism for finding effective embeddings, we will be able to implement a software tool that utilizes graph transformations for Euler diagram generation.

Acknowledgements This work is supported by the UK EPSRC grants EP/E011160/1 and EP/E010393/1 for the Visualization with Euler Diagrams project.

References

[1] S. Chow and F. Ruskey. Drawing area-proportional Venn and Euler diagrams. In *Proceedings of Graph Drawing 2003*, *LNCS 2912*, pages 466–477. Springer, 2003.

[2] R. DeChiara, U. Erra, and V. Scarano. A system for virtual directories using Euler diagrams. In *Proceedings of Euler Diagrams 04*, *ENTCS 134*, pages 33–53, 2005.

[3] L. Euler. Lettres a une princesse dallemagne sur divers sujets de physique et de philosophie. *Letters*, 2:102–108, 1775.

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

[5] D. Harel. On visual formalisms. In J. Glasgow, N. H. Narayan, and B. Chandrasekaran, editors, *Diagrammatic Reasoning*, pages 235–271. MIT Press, 1998.

[6] P. Hayes, T. Eskridge, R. Saavedra, T. Reichherzer, M. Mehrotra, and D. Bobrovnikoff. Collaborative knowledge capture in ontologies. In *Proceedings of the 3rd International Conference on Knowledge Capture*, pages 99–106, 2005.

[7] S. Kent. Constraint diagrams: Visualizing invariants in object oriented modelling. In *Proceedings of OOPSLA97*, pages 327–341. ACM Press, 1997.

[8] H. Kestler, A. Muller, T. Gress, and M. Buchholz. Generalized Venn diagrams: A new method for visualizing complex genetic set relations. *Journal of Bioinformatics*, 21(8):1592–1595, 2005.

[9] S.-K. Kim and D. Carrington. Visualization of formal specifications. In *6th Asia Pacific Software Engineering Conference*, pages 102–109. IEEE, 1999.

[10] O. Lemon and I. Pratt. Spatial logic and the complexity of diagrammatic reasoning. *Machine GRAPHICS and VISION*, 6(1):89–108, 1997.

[11] P. Rodgers, L. Zhang, and A. Fish. General Euler diagram generation. In *International Conference on the Theory and Application of Diagrams*, Springer, pages 13–27, 2008.

[12] G. Stapleton, P. Rodgers, J. Howse, and J. Taylor. Properties of Euler diagrams. In *Proceedings of Layout of Software Engineering Diagrams*, pages 2–16. EASST, 2007.

[13] A. Verroust and M.-L. Viaud. Ensuring the drawability of Euler diagrams for up to eight sets. In *Proceedings of 3rd International Conference on the Theory and Application of Diagrams*, *LNAI 2980*, Springer, pages 128–141, 2004.