
DOI
http://doi.org/10.1109/TVCG.2010.28

Link to record in KAR
http://kar.kent.ac.uk/30780/

Document Version
UNSPECIFIED

Copyright & reuse
Content in the Kent Academic Repository is made available for research purposes. Unless otherwise stated all content is protected by copyright and in the absence of an open licence (eg Creative Commons), permissions for further reuse of content should be sought from the publisher, author or other copyright holder.

Versions of research
The version in the Kent Academic Repository may differ from the final published version. Users are advised to check http://kar.kent.ac.uk for the status of the paper. Users should always cite the published version of record.

Enquiries
For any further enquiries regarding the licence status of this document, please contact:
researchsupport@kent.ac.uk

If you believe this document infringes copyright then please contact the KAR admin team with the take-down information provided at http://kar.kent.ac.uk/contact.html
Inductively Generating Euler Diagrams

Gem Stapleton, Peter Rodgers, John Howse and Leishi Zhang

Abstract—Euler diagrams have a wide variety of uses, from information visualization to logical reasoning. In all of their application areas, the ability to automatically layout Euler diagrams brings considerable benefits. In this paper, we present a novel approach to Euler diagram generation. We develop certain graphs associated with Euler diagrams in order to allow curves to be added by finding cycles in these graphs. This permits us to build Euler diagrams inductively, adding one curve at a time. Our technique is adaptable, allowing the easy specification, and enforcement, of sets of wellformedness conditions; we present a series of results that identify properties of cycles that correspond to the wellformedness conditions. This improves upon other contributions towards the automated generation of Euler diagrams which implicitly assume some fixed set of wellformedness conditions must hold. In addition, unlike most of these other generation methods, our technique allows any abstract description to be drawn as an Euler diagram. To establish the utility of the approach, a prototype implementation has been developed.

Index Terms—Information visualization, diagram layout, diagram generation, Euler diagrams, 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. A range of diagrams are based on finite collections of (usually simple) closed curves; such a collection of curves is called an Euler diagram [1]. To illustrate, the Euler diagram in Fig. 1 contains three closed curves, P, Q and R, which represent collections of objects (sets); it asserts that P and Q are disjoint, and that R may intersect with either P or Q. Venn diagrams are Euler diagrams in which all intersections between the curves are present. The diagram in Fig. 1 is not a Venn diagram; for instance, the intersection between all three curves is not present.

![Fig. 1. An Euler diagram.](image)

Euler diagrams and their extensions have wide-ranging uses in the area of information visualization, such as [2], [3], [4], [5], [6], [7]. Various methods for automatically generating Euler diagrams have been developed, each concentrating on a particular class of Euler diagrams; for example, see [6], [8], [9], [10], [11], [12], [13]. The generation algorithms developed so far produce Euler diagrams that have certain sets of properties, sometimes called wellformedness conditions; these conditions will be detailed below.

Each generation method starts with an abstract description of the required diagram. Typically, an abstract description specifies which intersections occur between curves. For example, the abstract description for the diagram in Fig. 1 includes the information that labels P, Q and R are used, together with one set of labels for each set intersection: \( \{P\}, \{P,R\}, \{R\}, \{R,Q\}, \{Q\}, \emptyset \); the presence of \( \emptyset \) reflects the fact that there is a region of the diagram outside all of the curves. We may abuse notation and write \( P \) for \( \{P\} \), \( PR \) for \( \{P,R\} \) and so forth.

![Fig. 2. Generation using a dual graph.](image)

Some existing generation approaches, such as [14], [15], construct a so-called dual graph from the abstract description, which is embedded in the plane, and ‘wrap’ closed curves around the dual graph, as illustrated in Fig. 2 (explained below). Each set of curve labels in the abstract description gives rise to a node in the graph, labelled by that set. Initially, two nodes are joined by an edge when their labels have exactly one curve label in their symmetric difference. Subgraphs of the dual graph are then sought which are planar and have a plane embedding which satisfies certain conditions [14]; for space reasons we omit the details. Once an appropriate embedding of some subgraph of the dual has been found, a layout for each curve in the diagram is determined. Finding an appropriate embedding can involve considering many subgraphs, and many different embeddings for each subgraph. To illustrate the embedding...
method, for the abstract description $\{\emptyset, P, Q, S, PR, QS\}$, we draw one node for each required set intersection and join the nodes as described; note that the node with no label in Fig. 2 corresponds to $\emptyset$. To embed the Euler diagram, this method finds a closed curve for each of $P$, $Q$, $R$ and $S$ that encloses precisely the nodes of the graph that include that curve in their label. So, $P$ must enclose the nodes labelled $P$ and $PQ$. In this simple example, the actual dual graph had an appropriate embedding.

In this paper, we present a novel approach to generation. We generate Euler diagrams inductively, adding one curve at a time. This is a more intuitive generation method, since it matches how people typically draw Euler diagrams [17] but prior to our work, no one has developed techniques that allow the construction for Venn diagrams [17] but prior to our approach to Euler diagram generation can be seen as extending the construction Venn provided in his original approach to Euler diagram generation can be seen as extending the construction Venn provided in his original paper, where he described how to add curves to Venn diagrams [16]. Edwards also developed an inductive construction for Venn diagrams [17] but prior to our work, no one has developed techniques that allow the inductive construction of Euler diagrams.

In order to add a curve, $c$, to an existing layout, we create a graph that is used to determine how $c$ is placed in the diagram. The manner in which we use this graph determines the wellformedness conditions that the generated Euler diagram satisfies. Hence, users can select the wellformedness conditions they wish to impose on the layout of the required diagram. This level of flexibility in imposing chosen wellformedness conditions is not incorporated into previously developed generation algorithms. In addition to contributing to the general Euler diagram generation problem, our approach is particularly advantageous in any situation where we wish to modify a diagram by adding a curve and maintain the existing layout; this type of situation occurs in reasoning contexts such as [18], [19], for example.

Section 2 overviews the syntax of Euler diagrams and other necessary background material. Abstractions of Euler diagrams are detailed in section 3. Section 4 presents the theory required in order to take an abstract description and decompose it into a sequence of abstract descriptions that reflects our inductive generation approach. In section 5, we define some graphs that allow us to add curves by finding appropriate cycles in them. Section 6 describes how we use a cycle to add a curve. In section 7, we present a series of results that show how to produce layouts that satisfy certain wellformedness conditions. Section 8 extends the techniques of the previous sections so that we can ensure any abstract description can be embedded. Finally, section 9 discusses a prototype implementation of the generation method and presents some output from the software. This paper significantly extends work presented in [20], which focuses on adding curves to diagrams that possess all five of the wellformedness conditions detailed below. In addition, we refer the reader to an appendix that accompanies this paper [21]; the appendix includes many examples to illustrate concepts developed throughout the paper along with proofs of most of the results.

2 Euler Diagrams

We now overview a formalization of Euler diagrams. Moreover, we also describe various concepts that will be required throughout the paper, in particular various wellformedness conditions, some associated graphs, and atomic diagrams. As stated above, an Euler diagram is a set of closed curves drawn in the plane. We assume that each curve has a label chosen from some fixed set of labels, $\mathcal{L}$.

Definition 2.1: An Euler diagram is a pair, $d = \langle Curve, l \rangle$, where

1. $Curve$ is a finite collection of closed curves each with codomain $\mathbb{R}^2$, and
2. $l: Curve \rightarrow \mathcal{L}$ is an injective function squared is that returns the label of each curve.

For example, $d_1$ in Fig. 1 contains three curves labelled $P$, $Q$ and $R$. To be more precise, $d_1$ depicts the images of three simple (i.e. non-self-intersecting) closed curves. The closed curves essentially provide a partition of the plane into minimal regions; in this example, there are 6 minimal regions, such as that inside both $P$ and $R$ but outside $Q$. Every diagram has a minimal region that is outside all of its curves. In order to define minimal regions, we need access to the images of the functions that give rise to the curves. Given a function, $c: \mathbb{R}^2 \rightarrow \mathbb{R}$, we write $image(c)$ to denote the image (sometimes called the range) of $c$:

$$image(c) = \{ b \in B : \exists a \in A c(a) = b \}.$$

Definition 2.2: A minimal region of an Euler diagram $d = \langle Curve, l \rangle$ is a connected component of

$$\mathbb{R}^2 \setminus \bigcup_{c \in Curve} image(c).$$

It is important to be able to identify the interior of closed curves. A point, $p \in \mathbb{R}^2 - image(c)$, is interior to a closed curve, $c$, if and only if the winding number of $c$ around $p$ is odd; see [22] for more details. Another important concept is that of zones.

Definition 2.3: A zone in an Euler diagram $d = \langle Curve, l \rangle$ is a non-empty set of minimal regions that can be described as being interior to certain curves (possibly no curves) and exterior to the remaining curves.

In Fig. 3, $d_2$ has 8 zones (and ten minimal regions) and $d_3$ has 6 zones (each of which is a minimal region). For example, in $d_2$, the zone interior to the curve labelled $Q$ only consists of two minimal regions. Minimal regions are purely a topological notion, and reflect a property of

1. Recall, a closed curve in the plane is a continuous function, $c: [a, b] \rightarrow \mathbb{R}^2$, where $c(a) = c(b)$.
2. An injective function, $l$, has the property that if $l(x) = l(y)$ then $x = y$. 

the drawn diagram. A zone, however, is a set of minimal regions that is taken to represent the intersection of sets represented by the curves that contain that zone, less the union of the sets represented by the curves that do not contain that zone. Zones are important since Euler diagrams represent these set intersections. Thus, a zone matches a semantic concept, unlike minimal regions.

Euler diagrams may possess certain properties, frequently called wellformedness conditions.

**Definition 2.4:** Given an Euler diagram \( d = (\text{Curve}, l) \), the following are properties that \( d \) may possess.

1. If all of the curves in \( \text{Curve} \) are simple then \( d \) possesses the **simplicity property**.
2. If no pair of curves in \( \text{Curve} \) run concurrently then \( d \) possesses the **no concurrency property**.
3. If there are no triple points of intersection between the curves in \( \text{Curve} \) then \( d \) possesses the **no concurrency property**.
4. If whenever two curves in \( \text{Curve} \) intersect, they cross then \( d \) possesses the **crossings property**.
5. If each zone in \( d \) is connected (i.e. consists of exactly one minimal region) then \( d \) possesses the **connected zones property**.

Formalizations of these properties can be found in [22]. To illustrate, in Fig. 3, \( d_2 \) possesses the simplicity and no concurrency properties but has a triple point (where \( P, Q \) and \( R \) intersect), two curves that do not cross where they intersect (where \( R \) and \( S \) intersect) and disconnected zones (such as that inside just \( R \)). The diagram \( d_3 \) possesses all five properties. Note that any diagram that possess the crossings property also possess the no concurrency property. Existing generation algorithms produce diagrams that possess specific subsets of these properties, in part for reasons of interpretability. The generation algorithm in [14], for example, draws only diagrams that satisfy all of these wellformedness conditions.

The concept of **nesting** in diagrams is of particular importance in automated layout. The (images of the) curves in an Euler diagram form connected components of \( \mathbb{R}^2 \). The diagram \( d_3 \) in Fig. 3, for example, consists of three components (the two curves labelled \( P \) and \( Q \), the curve labelled \( R \), and the curve labelled \( S \)) and is said to be nested. Intuitively, when generating Euler diagrams, we can break the problem down into one where each component of a nested diagram is generated independently, with the layouts subsequently merged in order to produce the required diagram.

**Definition 2.5:** Let \( d = (\text{Curve}, l) \) be an Euler diagram.

If \( \bigcup_{c \in \text{Curve}} \text{image}(c) \) consists of more than one connected subset of \( \mathbb{R}^2 \) then \( d \) is **nested**, otherwise \( d \) is **atomic** [23].

It is possible to identify when a diagram descriptions can be embedded in a nested manner and also to identify its atomic components prior to generation [23]; more details are given below.

Euler diagrams are associated with various graphs, some of which play an instrumental role in their automated layout; see [8], [14] for more details. In this paper, we are interested in two of these associated graphs. First, we can take an Euler diagram and construct its **Euler graph** which, roughly speaking, has a vertex at each point where two curves meet and the edges are the curve segments that connect the vertices; the Euler graph of \( d_1 \) in Fig. 1 is \( \text{EG}(d_1) \) in Fig. 4.

As a special case, any atomic component containing a single curve has exactly one vertex placed on that curve. The Euler graph was defined in [8], but the definition relies on certain wellformedness conditions holding. We extend the definition to the general case.

**Definition 2.6:** An **Euler graph** of Euler diagram \( d = (\text{Curve}, l) \) is a plane graph, denoted \( \text{EG}(d) \), whose embedded edges and vertices have image \( \bigcup_{c \in \text{Curve}} \text{image}(c) \) and \( \text{EG}(d) \) has a minimal number of vertices out of all the graphs to which it is homeomorphic (i.e. \( \text{EG}(d) \) has no unnecessary vertices of degree two).

Each face of the Euler graph is a minimal region of the Euler diagram. From the Euler graph, we can construct an **Euler graph dual** which is simply a dual graph of the Euler graph which we denote by \( \text{ED}(d) \). An Euler graph dual of \( d_1 \) in Fig. 1 is shown in Fig. 4. We note that Euler graphs and Euler graph duals are embedded in \( \mathbb{R}^2 \) and are plane. Given a vertex, \( v \), in the Euler graph dual, we write \( z(v) \) to mean the zone of \( d \) in which \( v \) is embedded. We will talk about the images of the edges and vertices of these embedded graphs as simply the edges and vertices respectively. Later, we will define further graphs, which are also embedded in \( \mathbb{R}^2 \), associated with Euler diagrams and again blur the distinction between the edges (vertices) and the embedding of those edges (vertices).

### 3 Diagram Descriptions

In order to generate an Euler diagram, \( d \), we start with an abstract description of \( d \). To illustrate, \( d_1 \) in Fig. 1 can be described as having three curves, \( P, Q \) and \( R \). These curves divide the plane in such a manner that
there are six zones present. Each zone can be described as being inside certain curves and outside the remaining curves. For instance, there is one zone inside $P$ only and another zone inside precisely $P$ and $R$. Thus, each zone can be described by the labels of the curves that the zone is inside. Note that there is always a zone outside all of the curves, which includes the unbounded minimal region (the unbounded face of the Euler graph), that is described by the empty set of labels, $\emptyset$; this is reflected in abstract descriptions.

Definition 3.1: An abstract description, $D$, is a pair, $(L, Z)$ where $L$ is a subset of $E$ (i.e. all of the labels in $D$ are chosen from the set $E$) and $Z \subseteq PL$ such that $\emptyset \in Z$. Elements of $Z$ are called abstract zones (or, simply, zones). Given $D = (L, Z)$, we define $L(D) = L$ and $Z(D) = Z$.

In Fig. 1, $d_1$ has abstract description $L = \{P, Q, R\}$ and $Z = \{\emptyset, \{P\}, \{P, R\}, \{R\}, \{Q, R\}, \{Q\}\}$.

Definition 3.2: Given an Euler diagram $d = (\text{Curve}, l)$, we map $d$ to the abstract description $\text{abstract}(d) = (\text{image}(l), Z)$, called the abstraction of $d$, where $Z$ contains exactly one abstract zone for each zone in $d$; in particular, given a zone, $z$, in $d$, the set $Z$ contains the abstract zone

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

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

The notions of being nested and atomic can be defined on abstract descriptions.

Definition 3.3: An abstract description $D = (L, Z)$ is nested if there exist abstract descriptions, $D_1 = (L_1, Z_1)$ and $D_2 = (L_2, Z_2)$, and a zone $z \in Z$ such that

1. $L_1$ and $L_2$ are both non-empty and form a partition of $L$, and
2. $Z_1 \cap \{z \cup z_2 : z_2 \in Z_2\} = \{z\}$ and $Z = Z_1 \cup \{z \cup z_2 : z_2 \in Z_2\}$.

If $D$ is not nested then $D$ is atomic [23].

Importantly, every atomic (nested) abstract description is the abstraction of some atomic (nested) Euler diagram [23]. However, atomic Euler diagrams may not have atomic abstractions, such as $d_3$ in Fig. 3. Any atomic diagram that has a nested abstraction can be redrawn in a nested manner. Finally, any abstraction can be drawn as an atomic Euler diagram.

4 Abstract Description Decomposition

The generation problem can be summarized as ‘given an abstract description, $D$, find an Euler diagram, $d$, such that $\text{abstract}(d) = D$ and $d$ satisfies some specified wellformedness conditions’. Our inductive approach will add curves successively until the generated Euler diagram has the specified abstract description. The manner in which we add the curves at each stage will be determined by the wellformedness conditions that have been selected.

3. Recall, the unbounded face, $f$, of a graph is that for which there is no disc of finite radius that encloses $f$.

To use an inductive generation approach, we need to know how to decompose an abstract description, $D$, into a sequences of abstract descriptions, $(D_0, D_1, ..., D_n)$ where $D_0$ contains no labels, $D_{n-1}$ is obtained from $D_0$ by removing a label, and $D_n = D$; the process is illustrated in Fig 5. It may be counterintuitive that we have written this sequence as $(D_0, D_1, ..., D_n)$ rather than $(D = D_n, D_{n-1}, ..., D_0)$; we choose to write $(D_0, D_1, ..., D_n)$ since our generation problem will start by finding an embedding of $D_0$ (which contains no curves), then $D_1$ (which contains 1 curve) and so forth, ending up with an embedding of $D_n = D$.

Definition 4.1: Given an abstract description, $D = (L, Z)$, and $\lambda \in L$, we define $D - \lambda$ to be $D - \lambda = (L - \{\lambda\}, \{z - \{\lambda\} : z \in Z\})$.

Definition 4.2: Given an abstract description, $D = (L, Z)$, a decomposition of $D$ is a sequence, $\text{dec}(D) = (D_0, D_1, ..., D_n)$ where each $D_i$ ($0 < i \leq n$) is obtained from $D_i$ by the removal of some label, $\lambda_i$, from $D_i$. If $D_0$ contains no labels then $\text{dec}(D)$ is a total decomposition.

The notion of a decomposition is similar to an abstraction of Euler diagrams developed in [24]. The technique we develop to add a curve to an Euler diagram, $d$, assumes that $d$ is atomic. Important for our generation approach, therefore, is that every abstract description has an atomic drawing.

To illustrate the generation process, we provide an example in Fig. 6, which produces an embedding of $D = (L, Z)$ where $L = \{P, Q\}$ and $Z = \{\emptyset, \{P\}, \{Q\}, \{P, Q\}\}$ given the total decomposition

$$\langle D_0 = (\emptyset, \emptyset), D_1 = (\{P\}, \emptyset), D_2 = D \rangle.$$
partially outside new curve (in this case, we say that the curve splits the zone). Thus, to obtain the required abstraction, we must know which zones are inside, which are outside and those that are to be split. In Fig. 6, the diagram embed(D1) has a curve P that splits the zone in embed(D0); the definition below captures the abstract level concept that corresponds to adding a curve, with the addition of P to D0 specified by in = {∅} and out = {∅}; since ∅ is split it can be considered as being both inside and outside the new curve.

Definition 4.3: Let D = (L, Z) be an abstract description. Let λ be a label in L − L and let in and out be two subsets of Z such that in ∪ out = Z and ∅ ∈ out. Then D + (λ, in, out) is defined to be

\[ D + (λ, in, out) = (L ∪ \{λ\}, Z_{in} ∪ Z_{out}) \]

where \( Z_{in} = \{z \cup \{λ\} : z ∈ in\} \) and \( Z_{out} = out \).

Lemma 4.1: Let D = (L, Z) be an abstract description and let λ ∈ L. Then \( D = (D − λ) + (λ, in, out) \) where

\[ in = \{z ∈ Z(D − λ) : z ∪ \{λ\} ∈ Z\} \]

and

\[ out = \{z ∈ Z(D − λ) : z ∈ Z\}. \]

The above lemma provides a framework for us to be able to describe how to add curves to Euler diagrams so that we construct embeddings of the abstract descriptions in any given total decomposition.

5 Graphs for Curve Addition

Our inductive generation method uses graph theoretic techniques and in this section we define various graphs. We modify the Euler graph dual and use this modified Euler dual along with the Euler graph to create a hybrid graph. It is the hybrid graph that we use for generation.

5.1 The Modified Euler Dual

A key insight to our approach of adding a curve is the observation that we can use cycles in an Euler graph dual to provide an embedding of a new curve. Recall, a cycle, C, in a graph G = (V, E) is a non-empty sequence of edges, \( C = (e_0, ..., e_n) \) in E, where no edge in E occurs more than once in C together with a sequence of vertices, \( (v_0, ..., v_n, v_{n+1}) \) such that \( v_0 = v_{n+1} \) and each edge, \( e_i \), in C is incident with \( v_i \) and \( v_{i+1} \); such a sequence of vertices is associated with C. The set of edges in C is denoted \( E(C) \) the set of vertices in the vertex sequence associated with C is denoted \( V(C) \).

As a simple example of our generation approach, to \( d_4 \) in Fig. 7 we may wish to add a curve that splits each zone. This can be done by finding a Hamiltonian cycle in an Euler graph dual, as illustrated in \( d_5 \).

If we wish to add a curve, S, to \( d_4 \) in Fig. 1 given in = \{∅, \{Q\}, \{Q, R\}, \{R\}\} and out = \{∅, \{P\}, \{P, R\}, \{R\}\} then no cycle in the Euler graph dual shown in Fig. 4 allows us to do so; see \( d_7 \) in Fig. 8 for how S may be added. In order to allow a curve such as S to be added we modify the Euler graph dual, as shown in \( d_8 \).

Intuitively, an Euler graph dual needs to be modified since an added curve may need to encode various minimal regions in the Euler diagram that are not enclosed by any cycle in an Euler graph dual. A given Euler graph dual does not necessarily reflect all isotopically different paths in \( \mathbb{R}^2 − V(EG(d)) \) that edges may take. However, in an atomic diagram, such different paths only exist between the vertex, \( v_i \), in \( ED(d) \) placed in the unbounded face, \( f_0 \), of \( EG(d) \) and those vertices adjacent to \( v_i \); this is because all other faces are topologically like a disc. Hence, the only modifications to \( ED(d) \) occur in \( f_0 \).

Definition 5.1: Let \( d = (Curve, l) \) be an atomic Euler diagram. A modified Euler dual of \( d \), denoted \( MED(d) \), is a plane graph obtained from the Euler graph dual of \( d \) by carrying out the following sequence of transformations:

1) for each edge, \( e \), incident with the vertex, \( v \), placed in the unbounded face, \( f_0 \), of \( EG(d) \) insert a new vertex of degree two onto \( e \) placed in \( f_0 \); the new vertex splits \( e \) into two edges in the obvious manner,
2) delete \( v \) along with all its incident edges; if this leaves any isolated vertices then delete those also,
3) add edges, embedded in \( f_0 \), connecting the newly inserted vertices (which have degree 1 after deleting \( v \)) so that the newly inserted vertices together with these new edges form a simple plane cycle\(^4\) that properly encloses the Euler graph.

5.2 The Hybrid Graph

As a further example of adding a curve to \( d_4 \) in Fig. 7, we may want to split all of the zones except that outside both curves. To do this, we cannot simply find a cycle in the modified Euler dual. The diagram \( d_9 \) shows \( MED(d_4) \) and how we might add R. Essentially, R is a cycle, therefore, is a cycle that is simple and in which no edges cross.

\(^4\) A simple cycle is one which does not pass through any vertex more than once (except the start and end vertex). A simple, plane cycle, therefore, is a cycle that is simple and in which no edges cross.
cycle in a graph that is formed by taking the Euler graph and the modified Euler dual and joining certain vertices with edges, as specified below. Intuitively, new curves can traverse edges of the modified Euler dual, edges in the Euler graph or the joining edges. The diagram $d_9$ in Fig. 9 shows how we can connect the modified Euler dual in $d_6$ to the Euler graph using additional edges (and two new vertices).

Given a plane graph, we can talk about faces and triangulations\(^5\), for example; $d_9$ is not a plane graph. Our final transformation adds vertices wherever two edges cross, as shown in $d_{10}$. An example of using this graph to add a curve labelled $R$ to $d_4$ which splits two zones, that inside just $P$ and that inside just $Q$, with the remaining zones outside $R$, can be seen in $d_{11}$.

![Fig. 9. Adding edges and inserting vertices to create the hybrid graph.](image)

We call the graph obtained by inserting these vertices and adding these edges the hybrid graph. It is this graph that we use to determine how to route new curves through a diagram. Each cycle in the hybrid graph is a path that can be followed by a new curve.

**Definition 5.2:** Let $d = (\text{Curve}, l)$ be an atomic Euler diagram. A hybrid graph for $d$, denoted $HG(d) = (V, E)$, is a plane graph obtained from $EG(d)$ and $MED(d)$ by carrying out the following sequence of transformations:

1) take the embeddings of $EG(d)$ and $MED(d)$ as one embedded graph, $G_1$, (i.e. union the vertex sets and union the edge sets),

2) for each edge, $e$, in $G_1$ that is in $MED(d)$ and completely embedded in the unbounded face, $f$, of $EG(d)$ insert a new vertex onto $e$; the new vertex splits $e$ into two edges in the obvious manner and we call the created graph $G_2$,

3) for each pair of edges, $e_1$ and $e_2$, in $G_2$, if $e_1$ and $e_2$ cross then insert a new vertex at the point where they cross; the new vertex splits each of $e_1$ and $e_2$ into two edges in the obvious manner, and we call the resulting graph $G_3$,

4) add edges to $G_3$ which are incident with a vertex in $MED(d)$ and a vertex in $EG(d)$ to create a graph, $G_4$, so that

a) all the new edges in $G_4$ are in the subgraph, $SG_4$, of $G_4$ generated by deleting the vertices of $G_4$ that are embedded in the unbounded face of $EG(d)$, and

b) $SG_4$ is triangulated except for its unbounded face,

5) add edges, $e$, to $G_4$, so that

a) $e$ is incident with a vertex in $EG(d)$,

b) $e$ is incident with a vertex in $G_2$ that is not in $MED(d)$ or in $EG(d)$, and
c) every vertex in $G_2$ that is not in $MED(d)$ or in $EG(d)$ is incident with exactly one new edge.

The resulting graph is $HG(d)$.

This process can be seen in Fig. 10, where the hybrid graph for $d_1$ in Fig. 1 is constructed. Note that, instead of adding vertices to create $G_3$, we could have just triangulated the whole graph (apart from the unbounded face) as in $G_4$. However, this would have meant that checking for crossings (see section 7) would have been more complex.

![Fig. 10. Constructing the hybrid graph.](image)

Given a hybrid graph for $d$, we partition the set of edges (vertices) as follows. Any edge (vertex) in the hybrid graph that arose from the Euler graph (i.e. those in black) is in the set $EulerEdges(HG(d))$ ($EulerVertices(HG(d))$). Any edge (vertex) in the hybrid graph that arose from the modified Euler dual (i.e. those in red) is in the set $DualEdges(HG(d))$ ($DualVertices(HG(d))$). The remaining edges (vertices) in the hybrid graph (i.e. those in green) are in the set $NewEdges(HG(d))$ ($NewVertices(HG(d))$). We call edges in the set $EulerEdges(HG(d))$ Euler edges and use similar terminology for elements of the other sets defined here.

## 6 ADDING CURVES

Any cycle in a hybrid graph can be used to add a curve to an Euler diagram (although the result need not be atomic). The manner in which this is done is captured by the following definition.

**Definition 6.1:** Let $d = (\text{Curve}, l)$ be an atomic Euler diagram with hybrid graph $HG(d)$. Let $C$ be a cycle in $HG(d)$ and let $\lambda$ be a label. Then $d$ extended by $C$ and $\lambda$ is defined to be an Euler diagram, denoted $d + (C, \lambda)$,
where \( d + (C, \lambda) = (\text{Curve} \cup \{e\}, I \cup \{(c, \lambda)\}) \) such that \( c \) is a closed curve, not in Curve, that traverses the cycle \( C \) and has label \( \lambda \).

A little more technically, the closed curve \( c \) to which the cycle \( C \) gives rise is formed by taking the union of the (injective) functions that gave rise to the embedded edges of which \( C \) consists (assuming any two consecutive embedded edges, \( e_i \) and \( e_{i+1} \), in \( C \) have domains of the form \([x, y]\) and \([y, z]\) where \( e_i(y) = e_{i+1}(y) \), but it is a trivial matter to change the domains if this fails to be so).

Observe that any curve added in this manner has a finite number (possibly zero) of self-intersection points, since no cycle contains an edge more than once. Curves, \( c \), with a finite number of self-intersections have nice properties with regard to winding numbers: intuitively, the winding number changes by ±1 each time we cross \( c \) at a point of non self-intersection and, as a consequence, we change from being inside \( c \) to outside \( c \) or vice versa.

In our generation process, we want to ensure that we find a cycle that gives rise to a diagram with some specified abstraction. We need to be able to identify whether dual vertices are inside the cycle or outside the cycle in order to know whether the zones in which they are placed will be inside or outside the new curve. Vertices in the modified Euler dual that are incident with some edge in the cycle, are embedded in zones that are split by the new curve. The concept of being inside a cycle will be defined by appealing to face-colouring.

**Lemma 6.1:** Let \( G = (V, E) \) be an Eulerian, plane graph. Then there is a face-colouring of \( G \) that uses at most two colours.

Given a cycle in a graph, \( G \), this cycle is essentially an Eulerian subgraph of \( G \). Therefore, we can use lemma 6.1 to define the inside and outside of a cycle, since we know we can two face-colour the cycle:

**Definition 6.2:** Let \( G = (V, E) \) be a plane graph. Let \( C \) be a cycle in \( G \) and denote the embedded subgraph of \( G \) containing precisely the edges in \( C \) and their incident vertices by \( SG(C) \). Further, suppose we have a face colouring of \( SG(C) \) that uses at most two colours. A vertex, \( v \in V - V(C) \), is **outside** \( C \) if it is embedded in a face of \( C \) that is coloured the same as the unbounded face of \( C \). Otherwise, \( v \in V - V(C) \) is **inside** \( C \).

In a hybrid graph, \( HG(d) \), the set of dual vertices that are outside (inside) some cycle \( C \) is denoted \( \text{outside}(C) \) (inside\( (C) \)). Further, we define

\[
\text{in}\text{Zones}(C) = \{\text{abstract}(z(v)) : v \in \text{inside}(C) \cup (V(C) \cap \text{DualVertices}(HG(d)))\}
\]

and

\[
\text{out}\text{Zones}(C) = \{\text{abstract}(z(v)) : v \in \text{outside}(C) \cup (V(C) \cap \text{DualVertices}(HG(d)))\}.
\]

We are now in a position to state a theorem that ties up the notion of adding a curve to an Euler diagram with its affect on the abstract description; the result follows immediately from the arguments above.

**Theorem 6.1:** Let \( d = (\text{Curve}, l) \) be an Euler diagram with hybrid graph \( HG(d) \). Let \( C \) be a cycle in \( HG(d) \) and let \( \lambda \) be a label that is not in \( d \), \( \lambda \notin \text{image}(l) \). Then \( \text{abstract}(d + (C, \lambda)) = \text{abstract}(d) + (\text{in}, \text{out}, \lambda) \) where

1) \( \text{in} = \text{in}\text{Zones}(C) \), and
2) \( \text{out} = \text{out}\text{Zones}(C) \).

**Theorem 6.2:** Let \( d = (\text{Curve}, l) \) be an Euler diagram with hybrid graph \( HG(d) \). Let \( C \) be a cycle in \( HG(d) \) that passes through at least one Euler vertex and let \( \lambda \) be a label that is not in \( d \), \( \lambda \notin \text{image}(l) \). Then \( d + (C, \lambda) \) is atomic.

So, when using our inductive generation approach, we seek cycles that pass through an Euler vertex.

### 7 Adding Curves Under Wellformedness Conditions

Embedding diagrams that possess certain wellformedness conditions has the potential to enhance readability. Unfortunately, it is known that not all abstract descriptions can be drawn when all five wellformedness conditions are imposed, for example, [25]. This means that, in order to find embeddings of some abstract descriptions, we sometimes need to allow certain wellformedness conditions to be broken. Moreover, users are likely to have different preferences about which wellformedness conditions they want to impose. Here, we consider (initially) each wellformedness condition and identify an equivalent condition on the cycle in the hybrid graph that will ensure the added curve does not break that wellformedness condition. The results are then generalized: for every subset of the wellformedness conditions, we identify an equivalent condition on the cycle in the hybrid graph that will ensure the added curve does not break any of the conditions in that set. Of course, such a cycle may not exist but the results allow us to seek appropriate cycles when adding curves under specified wellformedness conditions and identify when no such cycle exists (trivially, by an exhaustive search, for instance).

#### 7.1 Simplicity

The simplicity condition is very easy to enforce when adding a curve using a cycle: the cycle must not pass through any vertex more than once. Fig. 11 shows two cycles, highlighted in blue, in the hybrid graph for \( d_1 \). Fig. 1. The cycle in \( d_{12} \) gives rise to a non-simple curve, since it passes through a vertex more than once. The cycle in \( d_{13} \) gives rise to a simple curve since it does not pass through any vertex more than once. The diagrams obtained from \( d_1 \) by adding a curve using these cycles have the same abstraction.

**Definition 7.1:** Let \( d = (\text{Curve}, l) \) be an atomic Euler diagram with hybrid graph \( HG(d) \). Let \( C \) be a cycle in \( HG(d) \). Then \( C \) possesses the **simplicity property** whenever \( C \) is simple.

**Theorem 7.1:** Let \( d = (\text{Curve}, l) \) be an atomic Euler diagram with hybrid graph \( HG(d) \). Let \( C \) be a cycle in \( HG(d) \) and let \( \lambda \) be a label that is not in \( d \), \( \lambda \notin \text{image}(l) \). Then the curve added to \( d \) to give \( d + (C, \lambda) \) is simple if and only if \( C \) possesses the simplicity property.
The no concurrency condition requires the added curve not to run concurrently with any other curve and, moreover, not to run concurrently with itself. Since we are adding a curve to a diagram, \(d\), using a cycle in \(HG(d)\), by construction the curve does not run concurrently with itself: no cycle traverses an edge more than once. The existing curves in \(d\) give rise to the Euler edges in \(HG(d)\). Thus, our cycle must not include any Euler edges. To illustrate, \(d_{14}\) and \(d_{15}\), Fig. 12, both contain highlighted (blue) cycles that give rise to a diagram with the same abstraction when we use these cycles to add curves to \(d_1\) in Fig. 1. However, the blue cycle in \(d_{14}\) contains an Euler edge, so the added curve would run concurrently with, in this case, the curve labelled \(R\). The blue cycle in \(d_{15}\) does not contain an Euler edge and, therefore, does not give rise to any concurrency when a curve is added.

Fig. 11. The simplicity property.

**7.2 No Concurrency**

Definition 7.2: Let \(d = (\text{Curve}, l)\) be an atomic Euler diagram with hybrid graph \(HG(d)\). Let \(C\) be a cycle in \(HG(d)\). Then \(C\) possesses the no concurrency property whenever \(C\) does not contain any edges in \(\text{EulerEdges}(HG(d))\).

Theorem 7.2: Let \(d = (\text{Curve}, l)\) be an atomic Euler diagram with hybrid graph \(HG(d)\). Let \(C\) be a cycle in \(HG(d)\) and let \(\lambda\) be a label that is not in \(d\), \(\lambda \notin \text{image}(l)\). Then the curve added to \(d\) to give \(d + (C, \lambda)\) does not run concurrently with any curve in \(d\) or itself if and only if \(C\) possesses the no concurrency property.

In addition to labelling vertices with their multiplicity, we also label the edges. Given an edge, \(e\), in \(HG(d)\), we write \(\text{mul}(e, d)\) to mean the largest multiplicity of any point on \(e\) in \(d\). Note that using our inductive generation method, for any edge, \(e\), \(\text{mul}(e, d)\) is at most 1 for any incident vertex, \(v\), so this edge labelling is redundant. However, in a general Euler diagram, the multiplicity of a point on an Euler edge can be greater than that of any incident vertex; this occurs, for example, when a curve runs part way along an edge but does not meet a vertex.

Definition 7.3: Let \(d = (\text{Curve}, l)\) be an atomic Euler diagram with hybrid graph \(HG(d)\). Let \(C\) be a cycle in \(HG(d)\). Then \(C\) possesses the no triple points property whenever,

1) for any vertex, \(v\), in \(V(C)\), if \(\text{mul}(v, d) \leq 2\) then it is the case that \(\text{mul}(v, d)\) plus half the number of edges in \(C\) that are incident with \(v\) is at most two, and

2) for any edge, \(e\), in \(E(C)\), \(\text{mul}(e, d) \neq 2\).

Theorem 7.3: Let \(d = (\text{Curve}, l)\) be an atomic Euler diagram with hybrid graph \(HG(d)\). Let \(C\) be a cycle in \(HG(d)\) and let \(\lambda\) be a label that is not in \(d\), \(\lambda \notin \text{image}(l)\).
Then the curve added to \( d \) to give \( d + (C, \lambda) \) does not introduce any triple points if and only if \( C \) has no triple points property.

It may be the case that we are happy to allow triple points of intersection, but not quadruple points, for instance. An obvious generalization of the result above allows us to enforce an \( n \)-points wellformedness condition.

### 7.4 Crossings

There are various properties that our cycle must possess if it is to yield a curve that ensures the crossings property holds in the embedded diagram. First, we observe that any diagram that contains concurrency does not possess the crossings property. Thus, we cannot use Euler edges in our cycle when requiring only crossings. Second, suppose that the cycle contains an edge, \( e \), that is incident with an Euler vertex, \( v \), (\( e \) must be a new edge, since it cannot be an Euler edge). Then the curve added to the cycle in the cycle (which must also be a new edge) must ensure that the cycle crosses all of the curves that give rise to Euler edges incident with \( v \). In Fig. 14, \( d_{18} \) highlights a cycle (in blue) that creates a non-crossing point of intersection with \( P \) whereas that in \( d_{19} \) crosses the curves \( P \) and \( Q \) at each point it intersects them.

![Fig. 14. The crossings property.](image)

Fig. 15. Identifying crossings.

The notion of a crossing can be captured relatively straightforwardly: the cycle, when passing through an Euler vertex, \( v \), must have exactly half of the Euler edges incident with \( v \) on one side of it, as illustrated in Fig. 15 where the blue line segments indicate part of a cycle passing through an Euler vertex. A pair of consecutive edges, \( e_1 \) and \( e_2 \), in a cycle, therefore, gives rise to a two way partition of the edges, excluding \( e_1 \) and \( e_2 \), incident with the vertex \( v \) that joins \( e_1 \) and \( e_2 \). We denote the two sets in this partition by \( E_1(e_1, e_2, v) \) and \( E_2(e_1, e_2, v) \); thus, for crossings we require

\[
|E_1(e_1, e_2, v) \cap \text{EulerEdges}(HG(d))| = |E_2(e_1, e_2, v) \cap \text{EulerEdges}(HG(d))|
\]

for every pair of consecutive edges \( e_1 \) and \( e_2 \) in \( C \) that are incident with an Euler vertex \( v \). In Fig. 15, the lefthand illustration gives, denoting the set of Euler edges by \( EE \),

\[
|E_1(e_1, e_2, v) \cap EE| = |E_2(e_1, e_2, v) \cap EE| = 2 \quad \text{whereas in the righthand illustration } |E_1(e_1, e_2, v) \cap EE| = 1 \quad \text{and } |E_2(e_1, e_2, v) \cap EE| = 3.
\]

We must also ensure that the new curve does not create a non-crossing point of intersection with itself.

**Definition 7.4:** Let \( d = (\text{Curve}, l) \) be an atomic Euler diagram with hybrid graph \( HG(d) \). Let \( C = (e_0, ..., e_n) \) be a cycle in \( HG(d) \) with associated vertex sequence \((v_0, ..., v_n, v_0)\). Then \( C \) possesses the **crossings property** if and only if

1) \( C \) does not contain any edges that are in \( \text{EulerEdges}(HG(d)) \),

2) for any pair of consecutive edges, \( e_i \) and \( e_{i+1} \) in \( C \)

\[
|E_1(e_i, e_{i+1}, v_{i+1}) \cap \text{EulerEdges}(HG(d))| = |E_2(e_i, e_{i+1}, v_{i+1}) \cap \text{EulerEdges}(HG(d))|
\]

and

\[
|E_1(e_i, e_{i+1}, v_{i+1}) \cap E(C)| = |E_2(e_i, e_{i+1}, v_{i+1}) \cap E(C)|
\]

where we take \( e_{n+1} = e_0 \).

**Theorem 7.4:** Let \( d = (\text{Curve}, l) \) be an atomic Euler diagram that possesses the crossings property with hybrid graph \( HG(d) \). Let \( \lambda \) be a label that is not in \( d \), \( \lambda \notin \text{image}(l) \). Then \( d + (C, \lambda) \) possesses the crossings property if and only if \( C \) possesses the crossings property.

### 7.5 Connected Zones

Our final wellformedness condition is that of connected zones and it is linked to when we split a zone. Theorem 6.2 tells us that a zone, \( z \), is split by the new curve if there is a vertex incident with some edge in the cycle \( C \), embedded in some minimal region of \( z \). For example, this can be seen in Fig. 12 and Fig. 13. If a curve that splits a zone passes through that zone more than once then the zone will become disconnected. Fig. 12 and Fig. 13 each contain one diagram with a blue cycle that creates disconnected zones. The following lemma states the conditions under which a zone is split by a new curve given the assumption that all zones are connected in the original diagram.

**Lemma 7.1:** Let \( d = (\text{Curve}, l) \) be an atomic Euler diagram that possesses the connected zones property with abstraction \( \text{abstract}(d) = (\text{image}(l), Z) \). Let \( \text{in} \subseteq Z \) and \( \text{out} \subseteq Z \) such that \( \text{in} \cup \text{out} = Z \) and \( \emptyset \in \text{out} \). Let \( HG(d) \) be a hybrid graph of \( d \) and \( C \) be a cycle in \( HG(d) \). If \( \text{abstract}(d + (C, \lambda)) = \text{abstract}(d + (\text{in}, \text{out}, \lambda)) \) then

\[
in \cap \text{out} = \{\text{abstract}(z(v)) : v \in V(C) \cap \text{DualVertices}(HG(d))\}.
\]
Thus, to split a zone, the cycle must pass through that zone. If the curve passes through that zone more than once, then the zone becomes disconnected. In a slightly more general sense, we can think of a minimal region becoming disconnected if the cycle passes through that minimal region more than once.

**Definition 7.5:** Let \( d = (\text{Curve}, l) \) be an atomic Euler diagram with hybrid graph \( HG(d) \). Let \( C = (e_0, ..., e_n) \) be a cycle in \( HG(d) \) with associated vertex sequence \( (v_0, v_1, ..., v_n, v_{n+1}) \). Then \( C \) possesses the **connected minimal regions property** if and only if for every proper subsequence \( (v_i, v_{i+1}, ..., v_{i+j}) \) of \( (v_0, v_1, ..., v_n, v_{n+1}) \) where \( v_i \) and \( v_{i+j} \) are embedded in the same minimal region, \( m_s \) of \( d \), all of the vertices in the subsequences are also embedded in \( m_s \).

Intuitively, the definition says that a cycle which possesses the connected minimal regions property passes through each minimal region at most once and, hence, does not split that minimal region into more than two pieces. If all of the zones in a diagram are connected then the connected minimal regions property ensures that the added curve does not create any disconnected zones.

**Theorem 7.5:** Let \( d = (\text{Curve}, l) \) be an atomic Euler diagram that possesses the connected minimal regions property and has hybrid graph \( HG(d) \). Let \( C = (e_0, ..., e_n) \) be a cycle in \( HG(d) \). Let \( \lambda \) be a label that is not in \( d \), \( \lambda \notin \text{image}(l) \). Then \( d + (C, \lambda) \) possesses the connected zones property if and only if \( C \) possesses the connected minimal regions property.

**Theorem 7.6:** Let \( d = (\text{Curve}, l) \) be an atomic Euler diagram with hybrid graph \( HG(d) \). Let \( C = (e_0, ..., e_n) \) be a cycle in \( HG(d) \). Let \( \lambda \) be a label that is not in \( d \), \( \lambda \notin \text{image}(l) \). Then \( d + (C, \lambda) \) possesses the connected zones property if and only if

1. \( C \) possesses the connected minimal regions property,
2. each zone in \( d \) consists of at most two minimal regions,
3. for any zone in \( d \) that consists of two minimal regions, \( m_1 \) and \( m_2 \), the dual vertex embedded in one of \( m_1 \) and \( m_2 \) is inside \( C \) and the dual vertex embedded in the other is outside \( C \).

The above theorem can be used, and generalized, to allow embedded diagrams to contain disconnected zones enroute to producing an embedding that does not have disconnected zones. Given a total decomposition, \( \text{dec}(D) = (D_0, ..., D_n) \), the embedding of \( D_i \) can have zones consisting of at most \( 2^{n-i} \) minimal regions if we wish to ensure that the embedding of \( D_n \) has connected zones. The fact that the connected zones condition can be broken enroute to producing an embedding that has connected zones distinguishes it from the other conditions: if we break any other wellformedness condition then that wellformedness condition is broken in any diagram that contains additional curves.

### 7.6 Collections of Wellformedness Conditions

The above results extend very straightforwardly to arbitrary collections of the wellformedness conditions. We denote the set of five wellformedness conditions by \( WFC \). For each wellformedness condition, \( w \), we denote its corresponding property on a cycle, \( C \), in \( HG(d) \) by \( w(C) \). For example, the simplicity wellformedness condition corresponds to the simplicity property on \( C \), \( \text{Simplicity}(C) \) and the connected zones wellformedness condition corresponds to the connected minimal regions condition on cycles. For a set of wellformedness conditions, \( W \subseteq WFC \), we define \( W(C) = \{ w(C) : w \in W \} \).

**Theorem 7.7:** Let \( d = (\text{Curve}, l) \) be an atomic Euler diagram that possesses a set of wellformedness conditions \( W \). Let \( C \) be a cycle in a hybrid graph, \( HG(d) \), for \( d \) and let \( \lambda \in L - \text{image}(l) \). Then \( d + (C, \lambda) \) possesses all of the wellformedness conditions in \( W \) if any only if \( C \) possesses all of the properties in \( W(C) \).

The problem of adding a curve when all wellformedness conditions are imposed reduces to seeking a cycle in the modified Euler dual. We say that a diagram is **completely wellformed** if it possesses all five wellformedness conditions. As an example, in Fig. 17, the blue cycle in the hybrid graph in \( d_{24} \) reduces to a cycle in the modified Euler dual, as shown in \( d_{25} \). Intuitively, in the completely wellformed case, our cycle cannot use Euler edges since this would create concurrency.
Consider such a maximal subset, \( C \subseteq G \) duplicate that edge; the resulting graph, \( we will produce one curve, \( e \) Eulerian, since each of the two cycles that include \( e \) are Eulerian graphs. Given an Eulerian cycle, \( (e_0, \ldots, e_n) \), in \( G(C_1) \), the curve we introduce traverses that cycle, starting at the vertex \( v_0 \) incident with \( e_0 \), but each time we encounter a duplicate edge, the curve traverses the original edge, \( e \). Since any self-concurrency this curve possesses involves only double points of intersection, any point on one side of a concurrent line segment has a winding number the same parity as any point on the other side. Thus, points ‘inside’ either of the original cycles are precisely those inside the curve \( e_i \). To convert the curves produced at this stage into a single curve, we simply connect them with a minimal number of line segments. To ensure an atomic layout, at least one of the cycles must pass through an Euler vertex.

8 Ensuring Drawability

To ensure that all abstract descriptions can be drawn, we generalize our method for adding a curve; some diagrams cannot be drawn without curves that have self-concurrency but a cycle never gives rise to such a curve. For example, to \( d_1 \) in Fig. 1, we may want to add a curve that completely encloses the zones inside \( P, P \) and \( R, Q, \) and \( Q \) and \( R \), and in addition the remaining zones are all completely outside the curve; in this case, \( in = \{ \{ P \}, \{ Q \}, \{ Q, R \} \} \) and \( out = \emptyset \). However, no cycle in the hybrid graph (see Fig. 10) allows us to add a curve in the required manner.

Suppose we have an atomic diagram, \( d \), with abstraction \( abstract(d) = (l, Z) \). Let \( in \) and \( out \) be subsets of \( Z \) such that \( in \cup out = Z \) and \( \emptyset \in out \). To add a curve to \( d \) to give a diagram with abstraction \( abstract(d) + (in, out, L) \), where \( L \) is some label not in \( d \), find a set of cycles, \( C \), in \( H(d(d)) \) such that

1) ‘inside’ zones are inside some cycle:

\[ in = \bigcup_{C \in C} inZones(C), \]

2) ‘outside’ zones are outside all cycles:

\[ out = \bigcap_{C \in C} outZones(C), \]

3) for any two cycles, \( C_1 \) and \( C_2 \) in \( C \), there is no common vertex inside them, that is

\[ inside(C_1) \cap inside(C_2) = \emptyset. \]

We note that given any set of cycles that meets the conditions above ensures that any given edge in \( H(d) \) is in at most two cycles (this follows from the ‘disjoint interiors’ constraint).

To add an appropriate curve to the Euler diagram, first we partition the set, \( C \) into maximal subsets, \( C \), such that the subgraph, \( SG(C) \), containing exactly the cycles in \( C \) is connected; see Fig. 18, where three blue cycles in the hybrid graph are shown. For each of these subgraphs, we will produce one curve, \( e_i \), and then join these curves together. Consider such a maximal subset, \( C_i \). For each edge, \( e \), in \( SG(C_i) \) that is in more than one cycle in \( C_i \), we duplicate that edge; the resulting graph, \( G(C_i) \), is clearly Eulerian, since each of the two cycles that include \( e \)

Moreover, the cycle cannot use any new edges since this would create a triple point.

Fig. 17. Using the modified Euler dual to add a curve.

Fig. 18. Converting cycles in to a curve.

It is easy to justify the existence of a set of cycles that meet the criteria outlined above, as follows. For each abstract zone in \( in - out \), take the set of cycles around the faces of the minimal regions of the corresponding zone in \( d \). For each abstract zone in \( in \cap out \) choose a cycle that passes through a dual vertex, \( v_i \), embedded in the corresponding zone of \( d \) and some Euler edges around the face of the minimal region in which \( v \) is embedded. These cycles between them provide an appropriate \( C \). However, better diagram layouts may be achieved by using a minimal number of cycles or by minimizing the number of edges that are used in more than one cycle.

The results in section 7 that relate cycles to well-formedness conditions extend to this more general case in that each curve in \( C \) must possess the appropriate properties if a specified set of conditions are to hold. In addition, we must take into account how the cycles relate to each other. We note that the simplicity and
no concurrency properties cannot hold since the added curve violates these. Moreover, the crossings property cannot hold since the added curve intersects itself in a non-crossing manner at any point where a joining edge meets one of the cycles in \( C \). For the no triple points condition, we take into account how many times the cycles and the connecting edges pass through vertices. A zone, \( z \), becomes disconnected when at least two cycles, \( C_1 \) and \( C_2 \), in \( C \) contain edges, \( e_1 \) and \( e_2 \) respectively, that are embedded in \( z \) and are distinct. Moreover \( z \) becomes disconnected whenever a joining edge passes through it. Assuming all cycles in \( C \) have the connected zones property then it is sufficient to check that these other ways of disconnecting zones do not occur. Given the inevitability that the added curve will break at least three of the wellformedness conditions, we can choose to partially enforce them by ensuring that the cycles in \( C \) possess the required properties, for example.

9 Implementation

In this section, we describe a Java implementation of a system that takes an abstract description and outputs a diagram embedding. As noted in Section 2, the nested components of a diagram can be derived from the abstract description. Hence our system draws atomic diagrams which can then be joined together by placing nested components in the correct zone of the parent diagram. Given an Euler graph, we can derive the hybrid graph, and form a closed cycle that exactly meets the specification of zones that are inside and outside the new curve. Our software finds simple cycles, so only simple curves can be formed. The program can be found at www.eulerdiagrams.com/inductive/inductive.html.

Generating some restricted classes of diagrams can be performed in polynomial time [26]. However, current methods for generating diagrams from any abstract description take exponential time in the worst case [8]. Similarly our generation method has exponential time worst case performance relative to the number of vertices in the hybrid graph. This is because the problem of finding such cycles is, in general, NP-Complete [27], [28]. However optimizations can be derived: given a diagram, \( d = (\text{Curve}, l) \), with \( \text{abstract} (d) = (\text{image} (l), Z) \) and sets \( \text{in} \) and \( \text{out} \) such that \( \text{in} \cup \text{out} = Z \) and \( \emptyset \in \text{out} \), a cycle, \( C \), that adds an appropriate curve has the following properties

1) any connected zone, \( z \), in \( d \) that is to be partly inside and partly outside the new curve (i.e. to be split, so \( \text{abstract} (z) \in \text{in} \cap \text{out} \)) must have the corresponding Euler vertex in the vertex sequence associated with \( C \),

2) for any zone, \( z \), in \( d \) that is not to be split (i.e., \( \text{abstract} (z) \in (\text{in} \, \setminus \, \text{out}) \cup (\text{out} \, \setminus \, \text{in}) \)), it is the case that every Euler vertex embedded in \( z \) is not in the vertex sequence associated with \( C \),

3) any new vertex that is adjacent to two Euler vertices that are both embedded in zones, \( z_1 \) and \( z_2 \),

that are either to be properly inside or properly outside the new curve (i.e. \( \text{abstract}(z_1), \text{abstract}(z_2) \in \text{in} \, \setminus \, \text{out} \) or \( \text{abstract}(z_1), \text{abstract}(z_2) \in \text{out} \, \setminus \, \text{in} \) cannot be in \( C \).

These optimizations typically cut down the search space considerably, particularly because no edge in \( C \) can be incident with a vertex that cannot be in the vertex sequences associated with \( C \). In practice, diagrams with curves up to four curves can be embedded in reasonable time. Diagrams with more curves can be embedded if the number of split zones is not large. Further optimizations are possible in terms of the order in which curves are added in the total decomposition that we use for generation.

![Fig. 19](image_url)

**Fig. 19.** An automatically generated Euler diagram with its hybrid graph.

As an example of the output from the implementation, Fig. 19 shows an automatically generated hybrid graph of Venn-2 (the Venn diagram containing two curves, shown here in black). In addition to the restrictions on cycles to maximize wellformedness conditions (section 7), it is also possible to improve the layout using other criteria. In particular, to avoid meandering curve routes, we can choose an appropriate cycle that has minimal length. Fig. 20 shows a diagram obtained from Venn-2 (Fig. 19) by finding a minimal length cycle in the hybrid graph that results in adding a curve that splits each zone (thus creating Venn-3).

![Fig. 20](image_url)

**Fig. 20.** Adding a curve with minimal cycle length.
Fig. 21 shows an embedding of Venn-3 where we minimized the number of triple points, but kept the cycle length short as a second priority; the cycle used to add the curve here includes 10 hybrid graph edges, whereas that in Fig. 20 has only 9 edges. In both of these cases, it can be seen how the intricacy of cycles increases as each new curve added to the diagram. In our example, the curve labelled $a$ is added first, followed by $b$, and finally $c$. Of course, the hybrid graphs of these diagrams can be generated, and further diagrams produced, but have not been shown due to space restrictions.

Fig. 21. Adding a curve and minimizing triple points.

10 Conclusion

In this paper we have presented a novel, inductive approach to Euler diagram generation that uses graph theoretic techniques to add curves. A key advantage of our approach over previously developed techniques is the flexibility to generate diagrams under any collection of the typically enforced wellformedness conditions; this represents a significant advance for automated Euler diagram generation. The Java implementation of the theoretical results serves to illustrate the practical utility of the techniques.

The layout of the hybrid graph can have a profound impact on the effectiveness of the produced Euler diagrams. Force directed methods can be used to ensure a good layout of the hybrid graph, given various measures of ‘good’. We have yet to investigate what constitutes a good layout for hybrid graphs, but some work has been done in the case of the Euler diagrams themselves [29]. Using these empirical results, we can also use graph drawing techniques to improve the layout of the embedded Euler diagrams. In previous research we have taken a multi-criteria hill-climbing approach to improving the layout of Euler diagrams [30]. We have not yet applied these layout improvements to the diagrams in the paper, allowing us to focus the discussion on the new techniques, aiding explanation: if we had improved the layout of each diagram after each curve addition, it would be difficult to spot where the curve additions had occurred.

We also plan to identify further optimizations for our embedding algorithms. We suspect that further heuristics can be developed that narrow down the space through which we search when seeking an appropriate cycle, for example. Moreover, we will investigate graph transformation techniques that allow us to produce more efficiently a hybrid graph for $d + (C,L)$ given a hybrid graph for $d$. Currently, our implementation produces $HG(d + (C,L))$ from scratch, even though we have the hybrid graph for $d$; some subgraph of $HG(d)$ is a subgraph of $HG(d + (C,L))$, for example. Given that we know the cycle (or, more generally, set of cycles and joining edges) in $HG(d)$ that gave rise to $d + (C,L)$ we suspect that some efficient transformations are entirely feasible. Fig. 22 shows the strong similarity between the hybrid graphs before and after a curve addition, with the cycle highlighted in blue in $d_{26}$ giving rise to the new curve in $d_{27}$.

It is known that nested components can be drawn separately. A further extension includes allowing nested diagrams to have curves added to them: such techniques may yield more efficient generation algorithms. We briefly investigated such an extension in [20] for the completely wellformed case. The basic idea is to add one curve to each of the nested parts with which the new curve is required to intersect and then join up the new curves to create one curve, $c$, in the resulting diagram. Curves whose removal from a diagram increases the number of nested components are called disconnecting curves, the theory of which is developed in [31]. Alternatively, we note that every nested diagram can be transformed into an atomic diagram by moving the atomic components until they touch [8]. This means that each time we have a nested diagram to which we want to add a curve, we convert it in to an atomic diagram and then use the curve addition methods described above.

Whilst the focus of this paper is on curve addition, there are also cases where we wish to remove curves, or modify the zone set, after layout. When a curve is removed, this may result in the diagram breaking the connected zones condition whereas the original diagram may not have. One can sometimes perform transformations that restore this wellformedness condition; see [21] for an example. Indeed, given a diagram, $d$, the removal of any curve is guaranteed to result in a diagram that breaks a (not necessarily proper) subset of the well-
formedness conditions broken by \( d \), excluding the con-
ected zones condition.

**ACKNOWLEDGMENTS**

This work is supported by the UK EPSRC grants EP/E011160/1 and EP/E010393/1 for the Visualization with Euler Diagrams project. Thanks to Jean Flower, John Taylor, Andrew Fish and the anonymous reviewers their helpful comments.

**REFERENCES**


[5] P. Hayes, T. Eskridge, R. Saavedra, T. Reichherzer, M. Mebrotra, and D. Bobrovnikoff, “Collaborative knowledge capture in on-


abd D. Kane, B. Zeeberg, and J. Weinstein, “Vennmaster: Area-
proportional Euler diagrams for functional go analysis of microar-

[12] P. Simonetto and D. Auber, “Visualise undrawable Euler dia-

diagrams for up to eight sets,” in 3rd International Conference on the

International Conference on the Theory and Application of Diagrams.

anon,” in 5th International Conference on the Theory and Application of

[16] J. Venn, “On the diagrammatic and mechanical representation of
propositions and reasonings,” The London, Edinburgh and Dublin
Philosophical Magazine and Journal of Science, vol. 9, pp. 1–18, 1880.


[18] G. Stapleton, J. Mastroff, J. Flower, A. Fish, and J. Southern,
“Automated theorem proving in Euler diagrams systems,” Journal of

verify Euler/Venn homogeneous and Euler/Venn FOL hetero-
geneous rules of inference,” Software and System Modeling, vol. 3,

Euler diagrams from existing layouts,” in Layout of (Software)


[22] G. Stapleton, P. Rodgers, J. Howse, and J. Taylor, “Properties of
Euler diagrams,” in Layout of Software Engineering Diagrams.
EASST, 2007, pp. 2–16.

syntax, semantics and construction,” Software and Systems Modeling,


diagrammatic reasoning,” Machine GRAPHICS and VISION, vol. 6,

[26] H. Kestler, J. Messner, A. Müller, and R. Schuler,
“On the complexity of intersecting multiple circles for
graphical display,” University of Ulm, http://www.uni-

[27] C. Iwamoto and G. Toussaint, “Finding Hamiltonian circuits in
arrangements of Jordan curves is NP-complete,” in Information


[29] F. Benoy and P. Rodgers, “Evaluating the comprehension of
Euler diagrams,” in 11th International Conference on Information

diagrams,” in 7th International Conference on Information Visualisation.

[31] A. Fish and J. Flower, “Euler diagram decompostion,” in 5th
International Conference on the Theory and Application of Diagrams.

Gem Stapleton is a Senior Research Fellow, with interests including the
theory of diagrammatic logics and developing automated diagram layout
techniques. She has received the Best Paper Award at Diagrams 2004, was Runner-Up for the British Computer Society Distinguished Dissertation Award 2005, and was the only UK Finalist for the Cor Baayen Award 2006, presented by ERCIM to the most promising young researcher in Computer Science and Applied Mathematics. She was General Chair of Diagrams 2008.

Peter Rodgers is a Senior Lecturer and is head of the Computa-
tional Intelligence Research Group. His main research interests are in
diagrammatic visualization, including graph and Euler diagram layout
techniques. He has led several research projects supported by national
and international funding bodies. He sits on the program committee of
various international conferences.

John Howse is Professor of Mathematics and Computation and he
is leader of the Visual Modelling Research Group. His main research interests are diagrammatic reasoning and the development of visual
modelling languages. He is on the program committee for several
international conferences, was General Chair of Visual Languages and
Human-Centric Computing 2006, is on the steering committee for the
Diagrams conference series and was Program Chair for Diagrams 2008.
He received the Best Paper Award at Diagrams 2002.

Leishi Zhang is a Research Associate with a PhD in bioinformatics
visualization from the University of Brunel (funded by the EPSRC) and an MSc in computer science from University of Dundee. Her main research interests include information visualization, graph theory, artificial intelligence and data analysis. She has published her research in a number of international journals and conferences relating to the area of data analysis and visualization.
Inductively Generating Euler Diagrams: Appendices
Gem Stapleton, Peter Rodgers, John Howse and Leishi Zhang

1 INTRODUCTION

This document is an appendix to the paper entitled *Inductively Generating Euler Diagrams*, providing proofs of some results and many examples to illustrate further the concepts developed. Each section of the appendix contains additional details relating to the results presented under the same section heading in the paper.

2 EULER DIAGRAMS

This section, in the paper, introduces the concept of Euler diagrams and terminology with which we can discuss their properties. In order to define these properties, we often need to refer to the interior points of closed curves. Given a simple curve (one that does not self-intersect), it is easy to identify the interior since the curve partitions \( \mathbb{R}^2 \) into two pieces; the interior consists of the points in the bounded piece. However, in the general case it is not so straightforward to identify the interior. Thus, to define the interior of a curve we appeal to winding numbers: if the winding number of a curve, \( c \), around a point, \( p \), is odd then \( p \) is inside \( c \), otherwise \( p \) is outside \( c \).

\[ p_1 \quad p_2 \quad p_3 \]
\[ c_1 \quad c_2 \]

Fig. 1. Winding numbers are used to identify interior points.

**Example 2.1:** Fig. 1 shows two curves, \( c_1 \) and \( c_2 \), where the arrows indicate how we traverse the curve. The curve \( c_1 \) winds zero times around \( p_1 \), once around \( p_2 \) and twice around \( p_3 \). Thus, \( p_1 \) and \( p_2 \) are outside \( c_1 \) whereas \( p_2 \) is inside \( c_1 \). The curve \( c_2 \) contains \( q_1 \) and \( q_2 \), winding once and three times around these points respectively.

The point \( q_2 \) is outside \( c_2 \) since the winding number of \( c_2 \) around \( q_2 \) is two (and, therefore, not odd). Note that if we cross either \( c_1 \) or \( c_2 \) at a point where the respective curve does not self-intersect, the parity of the winding number changes. This observation allows us to identify interior points without knowing how we traverse the curve (or, more precisely, the function that gave rise to the drawing of the curve). To illustrate, in Fig. 2, we can draw a line from \( p_3 \) to a point, \( q \), in the unbounded region of \( \mathbb{R}^2 - \text{image}(c_1) \). Counting the number of times this line crosses \( c_1 \) is equivalent to computing the winding number of \( c_1 \) around \( p_3 \), provided this line does not pass through a point where \( c_1 \) self-intersects. In Fig. 2, we can see that the drawn line crosses \( c_1 \) twice, telling us that \( p_3 \) is outside \( c_1 \). This technique (of counting the number of times a line crosses a curve) to identify whether points are interior to a curve can be applied to any curve that intersects at a discrete set of points (i.e does not have any self-concurrency).

\[ q \quad p_3 \quad c_1 \]

Fig. 2. Identifying interior and exterior points from the curve image.

Fundamental to the work in the paper are five properties that Euler diagrams may possess: simplicity, no concurrency, no triple points, crossings, and connected zones.

\[ A \quad B \quad d_1 \]
\[ A \quad B \quad d_2 \]

Fig. 3. Non-simplicity and concurrency.

**Example 2.2:** The diagram \( d_1 \) in Fig. 3 possesses the no concurrency, no triple points and crossings properties.

---

* Gem Stapleton and John Howse are with Visual Modelling Group, University of Brighton, Brighton, UK. E-mail: {g.e.stapleton,john.howse}@brighton.ac.uk
* Peter Rodgers and Leishi Zhang are with University of Kent, Kent, UK. E-mail: {p.j.rodgers,l.zhang}@kent.ac.uk

Manuscript received September 10, 2008; revised September 8, 2009
However, $d_1$ does not possess the simplicity property: the curve labelled $B$ is not simple (it is a figure of 8). Also, $d_1$ does not possess the connected zones property: the zone inside just the curve $B$ consists of two minimal regions. The diagram $d_2$ possesses all properties except the concurrency property: the two curves share a concurrent line segment.

Also key to the work in the paper are the notions of the Euler graph and its dual, both of which are embedded in $\mathbb{R}^2$ and are plane. These graphs are illustrated in the paper, but we include further examples here to demonstrate various issues that are not reflected in the paper’s examples.

Fig. 4. Euler graphs.

Example 2.3: The Euler graphs of $d_1$ and $d_2$ in Fig. 3 are shown in Fig. 4 as $d_3$ and $d_4$ respectively. The graph $d_5$ is not an Euler graph of $d_1$ since it has too many vertices: it is homeomorphic to $d_3$ but has more vertices than $d_3$. The graph $d_6$ is not an Euler graph of $d_2$ since it does not have a vertex at the point where $A$ and $B$ meet near the bottom of the figure, thus making this graph non-plane (i.e., the edges intersect at some points where there are not vertices).

Fig. 5. Euler graph duals.

Example 2.4: Two different duals of the Euler graph $d_3$, Fig. 4, are shown in Fig. 5. The dual $d_7$ does not completely enclose $A$ whereas $d_8$ does.

3 DIAGRAM DESCRIPTIONS

Our generation approach focuses on embedding atomic abstract descriptions.

Example 3.1: The diagram $d_9$ in Fig. 6 has abstraction $D = \text{abstract}(d_9)$ where $L(D) = \{A, B, C\}$ and $Z(D) = \{\emptyset, \{A\}, \{A, B\}, \{A, C\}, \{A, B, C\}\}$. This abstract description is nested, established by taking $D_1$ with $L(D_1) = \{A\}$ and $Z(D_1) = \emptyset$ and $D_2$ with $L(D_2) = \{B, C\}$ and $Z(D_2) = \emptyset$. Clearly, $L(D_1)$ and $L(D_2)$ are both non-empty and form a partition of $L(D)$. Taking $z = \{A\}$, we see that

$$Z(D_1) \cap \{z \cup z_2 : z_2 \in Z(D_2)\} = \{\{A\}\} = \{z\}$$

and

$$Z(D_1) \cup \{z \cup z_2 : z_2 \in Z(D_2)\} = Z(D)$$

thus establishing that $D$ is indeed nested. However, $D_1$ and $D_2$ are both atomic.

As stated in the paper, every atomic (nested) abstract description is the abstraction of some atomic (nested) Euler diagram [1]. However, atomic Euler diagrams may not have atomic abstractions. Any atomic diagram that has a nested abstraction can be redrawn in a nested manner.

Example 3.2: The diagram $d_{10}$ in Fig. 6 is atomic but has a nested abstract description (its abstract description is the same as that for $d_9$).

4 ABSTRACT DESCRIPTION DECOMPOSITION

When we are presented with an abstract description that we wish to embed, we first decompose it into a sequence of abstract descriptions. To produce such a decomposition, we remove labels one at a time until there are none left.

Example 4.1: The diagram $d_{11}$ in Fig. 7 has abstraction $\text{abstract}(d_{11}) = D_{11}$ where $L(D_{11}) = \{A, B, C\}$ and $Z(D_{11}) = \emptyset$. Removing the label $C$ from $D_{11}$ gives the abstraction of $d_{12}$, with $L(D_{12}) = \{A, B\}$ and $Z(D_{12}) = \emptyset$, so $D_{11} - C = D_{12}$. Removing $B$ from $D_{12}$ gives the abstraction of $d_{13}$. The decomposition $\langle \text{abstract}(d_{16}), \text{abstract}(d_{15}), D_{12}, D_{11}\rangle$ is a total, atomic decomposition of $D_{11}$. The decomposition $\langle \text{abstract}(d_{16}), \text{abstract}(d_{15}), \text{abstract}(d_{14}), D_{11}\rangle$ is total but not atomic; $\text{abstract}(d_{14})$ is nested.
shown that
\[ Z(D) \subseteq Z((D - \lambda) + (\lambda, \text{in, out})). \]

Now let \( z \in Z((D - \lambda) + (\lambda, \text{in, out})) \). Then either \( z \in Z_{\text{in}} \) or \( z \in Z_{\text{out}} \). In the latter case, since \( Z_{\text{out}} = \text{out} \) and \( \text{out} \subseteq Z(D - \lambda) \), \( z \in Z(D - \lambda) \). Then, since
\[
\text{out} = \{ z \in Z(D - \lambda) : z \in Z \}
\]
we deduce that \( z \in Z(D) \). In the former case, \( \lambda \in z \), since \( z \) arose from \( Z_{\text{in}} \). This implies that \( z - \lambda \in \text{in} \) and it then follows that \( (z - \lambda) \cup \lambda = z \in Z(D) \), by the definition of the set \( \text{in} \) (stated in the lemma). Hence, in either case, \( z \in Z(D) \) and we have
\[ Z(D) \supseteq Z((D - \lambda) + (\lambda, \text{in, out})). \]

Hence
\[ Z(D) = Z((D - \lambda) + (\lambda, \text{in, out})) \]
as required. Therefore, \( D = (D - \lambda) + (\lambda, \text{in, out}) \).

\[ \square \]

5 Graphs for Curve Addition

Our inductive generation method finds cycles in the so-called hybrid graph that we define in the paper. Here, we provide further examples of the hybrid graph and its construction, together with justification that we can always construct such a graph. First, we consider the modified Euler dual from which the hybrid graph is built.

5.1 The Modified Euler Dual

We need to modify the Euler dual since it does not reflect all isotopically different paths that edges may take, relative to their end points (i.e. keeping the end points fixed where the vertices are embedded). Here, we are interested in paths through the plane, \( \mathbb{R}^2 \), minus the points where the vertices of the Euler graph lie.

Fig. 8. Isotopically different paths.

Example 5.1: Fig. 8 shows subgraphs of dual graphs of the Euler diagram \( d_5 \), Fig. 4. In each of \( d_{17} \) to \( d_{20} \), the red dual vertices are embedded in faces of the Euler graph; this is a first step towards constructing a dual graph.
Each red edge is an edge of a dual graph, connecting the same vertices in each diagram. However, the two red edges in \( d_{17} \) and \( d_{18} \) respectively take different paths through \( \mathbb{R}^2 \). It is possible to continuously transform the red edge in \( d_{17} \) until it passes through the precisely same points as the red edge in \( d_{18} \); such a transformation shows that the two lines are isotopic in \( \mathbb{R}^2 \) but, relative to their end points, necessarily passes through at least one black (Euler) vertex. However, if we restrict our notion of isotopy so that we cannot, during a transformation, pass through the points in \( \mathbb{R}^2 \) where the black Euler vertices are embedded then these two lines are not isotopically equivalent relative to their end points. The red edge in \( d_{19} \) and the red edge in \( d_{17} \) are isotopically equivalent in \( \mathbb{R}^2 \) minus the points where the black vertices are embedded. Similarly for the red edges in \( d_{18} \) and \( d_{20} \).

\[ \text{Fig. 9. Constructing the modified Euler dual.} \]

\textbf{Example 5.2:} The modified Euler dual is obtained from an Euler dual by carrying out a sequence of transformations. To illustrate, to the Euler dual shown in \( d_{21} \), Fig. 5, we first insert vertices on to the edges incident with the vertex in the unbounded face of the Euler graph; this results in \( d_{21} \), Fig. 9. The second stage in the construction process deletes the vertex, \( v \), embedded in the unbounded face of the Euler graph, along with its incident edges, resulting in \( d_{22} \). The final stage joins the vertices in \( d_{22} \) that are not in \( d_{20} \) (i.e. the newly inserted vertices) to give \( d_{23} \).

\textbf{Theorem 5.1:} Every atomic Euler diagram has a modified Euler dual.

\textbf{Proof:} First, we note that it is obvious that every atomic Euler diagram has an Euler graph and, therefore, has an Euler graph dual. Clearly, given any graph, we can insert vertices of degree two onto any edge, and split those edges into two new edges. Therefore, we can perform step 1 in the sequence of transformations that convert any Euler dual into a modified Euler dual. The fact that we can always perform step 2 (deleting vertices and incident edges) is equally obvious. For step 3, enclose both the graph that results after step 2 and the Euler graph by a simple closed curve that

1) passes through the points where the vertices in the unbounded face of the Euler graph are embedded, and
2) does not intersect any of the edges of either graph. Subdivide this curve in the obvious manner to give edges that connect these (newly inserted, at step 1) vertices, resulting in a simple plane cycle that properly enclosed the Euler graph, thus constructing \( MED(d) \). \( \square \)

5.2 The Hybrid Graph

\textbf{Example 5.3:} A hybrid graph of \( d_2 \), Fig. 3, is constructed in Fig. 10. First, the modified Euler dual and the Euler graph are taken as one graph, \( G_1 \). Second, we add vertices to \( G_1 \) that split the dual edges embedded in the unbounded face of \( G_1 \), to give \( G_2 \). The third step makes this graph plane, by inserting vertices wherever two edges cross, giving \( G_3 \). Next, we add triangulation edges to create \( G_4 \), but not in the unbounded face of \( EG(d) \). Finally, we add edges in this face, to give \( HG(d) \) as illustrated.

\[ \text{Fig. 10. Constructing the hybrid graph.} \]

\textbf{Theorem 5.2:} Every atomic Euler diagram has a hybrid graph.

\textbf{Proof:} Clearly, every atomic Euler diagram, \( d \), has an Euler graph \( EG(d) \) and we have shown that \( MED(d) \) also exists (theorem 5.1). Thus, we can obviously create \( G_1 \), \( G_2 \) and \( G_3 \) as described in the definition of the hybrid graph. We now show that we can construct \( G_4 \) from \( G_3 \). First, consider the graph obtained from \( G_3 \) by deleting all of the dual (red) vertices; call the resulting graph \( G'_3 \). The graph \( G'_3 \) has faces that are (essentially) the same as the faces of the Euler graph. Now, since \( d \) is atomic, the faces of \( EG(d) \) and, therefore, of \( G'_3 \) are all simply connected (except the unbounded face). Let \( f \) be such a face. Then \( f \) is bounded by edges of the graph. As we traverse the bounding edges of \( f \), we see that the vertices around \( f \) alternate between Euler (black)
vertices and red vertices; they alternate since there is exactly one red vertex on every Euler edge, because every Euler edge is crossed by a dual edge. Now, in $G_2$, there is exactly one red (red) vertex, $v$, embedded in $f$. Connecting $v$ to all of the Euler vertices around $f$ with new edges (as in the construction of $G_2$) therefore triangulates the face, since the new (red) vertices around $f$ are already incident with $v$ in $G_2$. Hence, we can construct $G_3$. Finally, we must show that we can turn $G_3$ into the hybrid graph.

Clearly, we can add edges that satisfy 5(a) and 5(b) of the construction process. Thus, we must just establish that we can achieve 5(c). Let $v$ be a (red) vertex in $G_2$ that is not in $G_1$. We want to ensure that we can add an edge incident with $v$ and incident with a (black) vertex in $EG(d)$. Now, since $v$ is inserted on an edge of $MED(d)$ that is embedded in the unbounded face of $EG(d)$, we know that $v$ is incident with vertices $v_1$ and $v_2$ of $MED(d)$. Moreover, we know that $v$ is next to two faces of $G_1$: its unbounded face and a bounded face, $f$. The vertices $v_1$ and $v_2$ are also next to $f$. Assume, first, that $v_1$ and $v_2$ are distinct. Studying the vertices as we traverse the edges around $f$, we see that at least one of them must be an Euler vertex, $v_3$ (since the vertices $v_1$ and $v_2$ arise from the Euler graph dual, whose edges each cross an Euler graph edge). Therefore, we can connect $v$ to $v_3$ in order to create $HG(d)$. Alternatively, $v_1$ and $v_2$ are not distinct. In this case, $v_1$ is incident with an edge that is a loop and properly encloses the Euler graph of $d$. Since the Euler graph has at least one vertex next to its unbounded face, we can connect $v$ to that vertex as required. Therefore, in either case, we can add an edge to connect $v$ to an Euler vertex. Since $v$ was an arbitrary vertex in $G_2$ but not in $EG(d)$ or $MED(d)$, we can construct $HG(d)$.

Thus, it is helpful to be able to identify whether a cycle will give rise to an appropriate curve without having to embed the curve itself. This motivates the definition of the interior and exterior of a cycle, defined in such a manner that it agrees with the interior and exterior of any curve that the cycle generates (as defined in the paper). The concept of being inside a cycle is defined by appealing to face-colouring.

![Fig. 12. Identifying the interior of cycles.](image)

**Example 6.2:** In Fig. 12, the cycle $C$ whose inside is shaded is used to add $D$ to $d_2$, Fig. 3. The inside of the curve $D$ is also shaded. The graph $G$ shows an alternative cycle in $HG(d_2)$. The bottom right diagram illustrates a two-face-colouring of this cycle, with the regions coloured yellow being outside the cycle and those coloured white being inside the cycle.

### 7 Adding Curves Under Wellformedness Conditions

Here, we present proofs for many of the results stated in this section of the paper.

#### 7.1 Simplicity

**Theorem 7.1:** Let $d = (Curve, I)$ be an atomic Euler diagram with hybrid graph $HG(d)$. Let $C$ be a cycle in $HG(d)$ and let $\lambda$ be a label that is not in $d$, $\lambda \notin image(I)$. Then the curve added to $d$ to give $d + (C, \lambda)$ is simple if and only if $C$ possesses the simplicity property.

**Proof:** Suppose, first, that the curve, $c$, added to $d$ to give $d + (C, \lambda)$ is simple. Then the cycle $C$ cannot have passed through any vertex more than once, for if it did then $c$ would self-intersect by construction. Therefore, $C$ is simple and, hence, possesses the simplicity property. Conversely, suppose that $C$ possesses the simplicity property. Then $C$ is simple, meaning that the curve, $c$, constructed from $C$ is also simple.

#### 7.2 No Concurrency

**Theorem 7.2:** Let $d = (Curve, I)$ be an atomic Euler diagram with hybrid graph $HG(d)$. Let $C$ be a cycle in
\( H G(d) \) and let \( \lambda \) be a label that is not in \( d, \lambda \notin \text{image}(l) \). Then the curve added to \( d \) to give \( d + (C, \lambda) \) does not run concurrently with any curve in \( d \) or itself if and only if \( d \) possesses the no concurrency property.

**Proof:** Suppose, first, that the curve \( c \) added to \( d \) to give \( d + (C, \lambda) \) does not run concurrently with any curve in \( d \) or itself. This means that any edge in \( C \) (i.e. precisely the edges that gave rise to the curve \( c \)) did not arise from an Euler edge (the black edges), for the presence of such an edge would mean that \( c \) ran concurrently with the curve(s) from which that Euler edge arose. Therefore, \( C \) possesses the no concurrency property.

Conversely, suppose that \( C \) possesses the no concurrency property. Then \( d \) does not introduce any Euler edges by definition. Therefore, \( c \) does not run concurrently with any other curve in \( d \). Moreover, \( c \) does not run self-concurrently, since this would imply that an edge occurred twice in \( C \) but then \( C \) would not be a cycle. Hence, \( c \) does not run concurrently with any curve in \( d \) or itself. \( \square \)

### 7.3 No Triple Points

In order to detect triple points when adding curves, we need to know the multiplicity of any existing points of intersection.

**Theorem 7.3:** Let \( d = (\text{Curve}, l) \) be an atomic Euler diagram with hybrid graph \( H G(d) \). Let \( C \) be a cycle in \( H G(d) \) and let \( \lambda \) be a label that is not in \( d, \lambda \notin \text{image}(l) \). Then the curve added to \( d \) to give \( d + (C, \lambda) \) does not introduce any triple points if and only if \( C \) possesses the no triple points property.

**Proof:** Suppose, first, that the curve, \( c \), added to \( d \) to give \( d + (C, \lambda) \) does not introduce any triple points. Let \( v \) be a vertex in \( V(C) \). Suppose that \( \text{mul}(v, d) \leq 2 \). Since \( c \) does not introduce any triple points, we know that \( \text{mul}(v, d + (C, \lambda)) \leq 2 \). By the construction of \( c \), this implies that the number of times \( C \) passes through \( v \) is at most \( 2 - \text{mul}(v, d) \). In other words, \( \text{mul}(v, d) \) plus half the number of edges in \( C \) is at most two. Let \( e \) be an edge in \( E(C) \). Then \( c \) traverses \( e \) exactly once, by construction. Therefore, the multiplicity of that edge increases by 1. Since \( c \) does not introduce any triple points, we know that \( \text{mul}(e, d) \neq 2 \) in \( d \). Therefore, \( C \) possesses the no triple points property.

Conversely, suppose that \( C \) possesses the no triple points property. We must show that \( c \) does not introduce any triple points. We proceed by contradiction. Let \( p \) be a point in \( \text{image}(c) \) such that \( p \) is a triple point in \( d + (C, \lambda) \) but not a triple point in \( d \) (i.e. we assume that \( c \) introduced a triple point). There are three cases.

1) First, \( p \) is a 0-point of intersection in \( d \) (is not in the image of any curve in \( d \). In this case, \( c \) passes through \( p \) at least three times. Suppose first that \( p \) is a point where a vertex, \( v \), of \( C \) is embedded. We know, therefore, that \( \text{mul}(v, d) = 0 \). Moreover, we deduce that \( C \) contains at least six edges incident with \( v \), since \( p \) becomes a triple point when \( c \) is added. Therefore, \( \text{mul}(v, d) \) plus half the number of edges in \( C \) incident with \( v \) is at least 3, implying that \( C \) does not possess the triple points property, which is a contradiction. Now suppose that \( p \) is a point on some edge, \( e \), in \( C \) but not where a vertex is embedded. Since \( p \) is a 0-point in \( d \), we know that \( \text{mul}(e, d) = 0 \). Therefore, \( c \) must pass along \( e \) at least three times. This implies, by the construction of \( c \), that \( C \) must contain \( e \) at least three times, but cycles contain edges at most once giving a contradiction. Hence, \( p \) is not a 0-point of intersection.

2) Second, suppose that \( p \) is a 1-point in \( d \). This is similar to case 1.

3) Finally, suppose that \( p \) is a 2-point in \( d \). The case when \( p \) is a point on which a vertex lies is similar to case 1. When \( p \) lies on an edge, we immediately see that \( e \) has multiplicity 2 (\( \text{mul}(e, d) = 2 \)) but this cannot be so since \( C \) possesses the no triple points property.

Hence if \( C \) possesses the no triple points property then \( c \) does not introduce any triple points. \( \square \)

### 7.4 Crossings

**Theorem 7.4:** Let \( d = (\text{Curve}, l) \) be an atomic Euler diagram that possesses the crossings property with hybrid graph \( H G(d) \). Let \( \lambda \) be a label that is not in \( d, \lambda \notin \text{image}(l) \). Then \( d + (C, \lambda) \) possesses the crossings property if and only if \( C \) possesses the crossings property.

**Proof:** Suppose, first, that \( d + (C, \lambda) \) possesses the crossings property. Let \( e \) be an edge in \( C \). We show that \( e \) is not an Euler edge. The curve, \( c \), added to \( d \) crosses all curves in \( d \) whenever it intersects them and, moreover, crosses itself at any point of self-intersection. This means that \( e \) does not run concurrently with any curve in \( d \) and, therefore, does not traverse any Euler edge, implying \( e \) is not an Euler edge.

Let \( e_i \) and \( e_{i+1} \) be consecutive edges in \( C \). Suppose that \( v_{i+1} \) is incident with an Euler edge. In \( d \), all of the curves that pass through \( v \) cross at \( v \) (since \( d \) possesses the crossings property). This implies that every curve that passes through \( v \) gives rise to two edges in the Euler graph (i.e. the number of Euler edges that pass through \( v \) is twice the number of curves that pass through \( v \)). Now, the curve \( c \) traverses \( e_i \), passes through \( v_{i+1} \) and then traverses \( e_{i+1} \). Let \( c' \) be a curve in \( d \) that passes through \( v_{i+1} \). Then one of the two Euler edges incident with \( v_{i+1} \) arising from this curve is in the set \( E_1(e_i, e_{i+1}, v_{i+1}) \) and the other is in \( E_2(e_i, e_{i+1}, v_{i+1}) \), or \( c \) would not cross \( c' \). From this, it follows that there are the same number of Euler edges in \( E_1(e_i, e_{i+1}, v_{i+1}) \) and \( E_2(e_i, e_{i+1}, v_{i+1}) \), as required for the crossings property to hold for \( C \). A similar argument shows that

\[ |E_1(e_i, e_{i+1}, v_{i+1}) \cap E(C)| = |E_2(e_i, e_{i+1}, v_{i+1}) \cap E(C)|. \]

Hence \( C \) possesses the crossings property.
Conversely, suppose that \( C \) possesses the crossings property. We must show that each time \( c \) intersects a curve in \( d \) it crosses that curve. Let \( p \) be a point where \( c \) intersects some curve, \( c' \), in \( d \). Then clearly \( p \) is not a point on an Euler edge, since \( C \) does not contain any Euler edges. Therefore, \( p \) is a point where an Euler vertex, \( v \), is embedded. Let \( e_i \) and \( e_{i+1} \) be consecutive edges in \( C \) where \( v_{i+1} = v \); these two edges form part of \( c \) that passes through \( p \). Suppose, for a contradiction, that \( c' \) does not cross \( c \) at \( p \). Then \( c' \) gives rise to two Euler edges, \( e_1 \) and \( e_2 \), that are both in either \( E_1(e_i, e_{i+1}, v_{i+1}) \) or \( E_2(e_i, e_{i+1}, v_{i+1}) \) where \( c \) traverses first \( e_i \) and then, immediately, traverses \( e_2 \). But then \( c' \) does not cross all curves in \( d \) at \( p \) contradicting the fact that \( d \) possesses the crossings property. Thus, \( c \) must cross all curves in \( d \) whenever it intersects them. The argument to show that each time \( c \) self-intersects, it crosses itself is similar. Hence \( d + (C, \lambda) \) possesses the crossings property.

Then \( d + (C, \lambda) \) possesses the connected zones property if and only if \( C \) possesses the connected minimal regions property.

**Proof:** Suppose, first, that \( d + (C, \lambda) \) possesses the connected zones property. Let \( \{v_i, v_{i+1}, \ldots, v_{i+j}\} \) be a proper subsequence of \( \{v_0, v_1, \ldots, v_n\} \). Let \( c \) be the curve added to \( d \) to give \( d + (C, \lambda) \). If \( c \) enters and leaves a zone twice (or more) as one traverses \( c \) then \( c \) would create disconnected zones. Since \( d + (C, \lambda) \) has connected zones, we deduce that \( c \) enters and leaves each zone at most once. Thus, if \( v_i \) is embedded in a minimal region, \( m_i \), and so is \( v_{i+j} \), all of the intermediate vertices must also be embedded in \( m \). Thus, \( C \) possesses the connected minimal regions property. The converse is similar. \( \Box \)

**Theorem 7.6:** Let \( d = (Curve, l) \) be an atomic Euler diagram with hybrid graph \( HG(d) \). Let \( C = (e_0, \ldots, e_n) \) be a cycle in \( HG(d) \). Let \( \lambda \) be a label that is not in \( d \), \( \lambda \notin image(l) \). Then \( d + (C, \lambda) \) possesses the connected zones property if and only if

1. \( C \) possesses the connected minimal regions property,
2. each zone in \( d \) consists of at most two minimal regions,
3. for any zone in \( d \) that consists of two minimal regions, \( m_1 \) and \( m_2 \), the dual vertex embedded in one of \( m_1 \) and \( m_2 \) is inside \( C \) and the dual vertex embedded in the other is outside \( C \).

The above theorem can be used, and generalized, to allow embedded diagrams to contain disconnected zones enroute to producing an embedding that does not have disconnected zones.

**7.6 Collections of Wellformedness Conditions**

**Theorem 7.7:** Let \( d = (Curve, l) \) be an atomic Euler diagram that possesses a set of wellformedness conditions \( W \). Let \( C \) be a cycle in a hybrid graph, \( HG(d) \), for \( d \) and let \( \lambda \in L - image(l) \). Then \( d + (C, \lambda) \) possesses all of the wellformedness conditions in \( W \) if any only if \( C \) possesses all of the properties in \( W(C) \).

**Proof:** This follows immediately from the previous theorems on wellformedness conditions. \( \Box \)

The problem of adding a curve when all wellformedness conditions are imposed reduces to seeking a cycle in the modified Euler dual.

**Theorem 7.8:** Let \( d = (Curve, l) \) be an atomic, completely wellformed Euler diagram with hybrid graph \( HG(d) \). Let \( abstract(d) = (l, Z) \) be an abstraction of \( d \) and let \( in, out \subseteq Z \) such that \( in \cup out = Z \) and \( \emptyset \in out \). Then there exists a cycle, \( C_1 \), in \( HG(d) \) such that \( d + (L, C_1) \) is completely wellformed and has abstraction \( abstract(d) + (\lambda, in, out) \) if and only if there exists a simple cycle, \( C_2 \) in \( MED(d) \) such that \( d + (L, C_2) \) has abstraction \( abstract(d) + (\lambda, in, out) \).

Thus, if we want to enforce all wellformedness conditions, the generation task can be reduced to finding appropriate simple cycles in the modified Euler dual. In fact, we have a stronger result than the one given above.
Theorem 7.9: Let \( d = (\text{Curve}, I) \) be an atomic, completely wellformed Euler diagram with hybrid graph \( HG(d) \). Let \( \text{abstract}(d) = (I, Z) \) be an abstraction of \( d \) and let \( \text{in}, \text{out} \subseteq Z \) such that \( \text{in} \cup \text{out} = Z \) and \( \emptyset \in \text{out} \). Then there exists a curve, \( c \), that is not in \( \text{Curve} \) such that \( d + (c, \lambda) \) is atomic, completely wellformed and has abstraction \( \text{abstract}(d) + (\lambda, \text{in}, \text{out}) \) if and only if either

1) there exists a plane, simple cycle, \( C \), in \( \text{MED}(d) \) such that
   a) \( C \) possesses the connected minimal regions property,
   b) the vertices in \( \text{MED}(d) \) that are embedded in zones whose abstractions are elements of the set \( \text{in} \cap \text{out} \) are exactly those in \( V(C) \),
   c) the vertices in \( \text{MED}(d) \) that are embedded in zones whose abstractions are elements of the set \( \text{in} - \text{out} \) are located inside \( C \), and
   d) the vertices in \( \text{MED}(d) \) that are embedded in zones whose abstractions are elements of the set \( \text{in} - \text{out} \) are located outside \( C \),

or

2) \( |\text{in}| = |\text{in} \cap \text{out}| = 2 \) and the two zones corresponding to the abstract zones that are elements of the set \( \text{in} \cap \text{out} \) are topologically adjacent.

Thus, to find a completely wellformed embedding, when using the modified Euler dual we know almost exactly the set of vertices through which our cycle, \( C \), must pass: \( C \) must pass through all vertices embedded in the zones that are to be split, except the zone, \( z \), outside all curves. In the case of \( z \), the cycle must pass through at least one vertex embedded in \( z \). In the special case of Venn diagrams, it was observed in [2] that adding curves corresponded to finding Hamiltonian cycles in some appropriate dual graph.

8 Ensuring Drawability

To ensure that we can draw any abstract description, we sometimes need to find a set of cycles with certain properties.

Example 8.1: Fig. 13 shows a diagram, \( d_{24} \), together with its hybrid graph, \( d_{25} \). There is no cycle in the hybrid graph that allows us to add a curve, labelled \( F \), that splits the zones

\[ \text{in} = \{\emptyset, \{A\}, \{A, B, E\}, \{B, E\}\}, \]

(and we take \( \text{out} = Z(\text{abstract}(d_{24})) \)). In this case, to add a curve in the required manner, we can find two cycles, \( C_1 \) and \( C_2 \), where \( C_1 \) splits \( \emptyset \) and \( \{A\} \) and \( C_2 \) splits \( \{A, B, E\} \) and \( \{B, E\} \); these cycles are shown in \( d_{26} \) and \( d_{27} \), respectively. These two cycles can then be joined by an edge (which we traverse twice) to give \( F \), shown in \( d_{28} \). For \( C_1 \), we have

\[ \text{inZones}(C_1) = \{\emptyset, \{A\}\}, \]

(i.e. no zones are properly inside \( C_1 \) and the two stated zones are split by \( C_1 \), and

\[ \text{outZones}(C_1) = Z(\text{abstract}(d_{24})) \]

(i.e. all zones in \( d_{24} \) give rise to an abstract zone in \( \text{outZones}(C_1) \) since each such zone is either properly outside \( C_1 \) or split by \( C_1 \)).

For \( C_2 \), we have

\[ \text{inZones}(C_2) = \{\{A, B, E\}, \{B, E\}\}, \]

and \( \text{outZones}(C_2) = Z(\text{abstract}(d_{24})) \).

We see that the zones that are in \( \text{in} \) are given by

\[ \text{in} = \text{inZones}(C_1) \cup \text{inZones}(C_2). \]

Moreover, the zones that in \( \text{out} \) are given by

\[ \text{out} = \text{outZones}(C_1) \cap \text{outZones}(C_2). \]

Example 8.2: Suppose we want to add a curve, \( G \), to \( d_{24} \), Fig. 13, where

\[ \text{in} = \text{abstract}(d_{24}), \]

and

\[ \text{out} = \{\emptyset\}. \]

The two cycles, \( C_3 \) and \( C_4 \) shown in \( d_{29} \) and \( d_{30} \), respectively, Fig. 14, between them have

1) \( \text{inZones}(C_3) \cup \text{inZones}(C_4) = \text{in} \), and

2) \( \text{outZones}(C_3) \cap \text{outZones}(C_4) = \text{out} \).

However, in this case, we also have \( \text{inside}(C_3) \cap \text{outside}(C_4) \neq \emptyset \), since there is a new (green) vertex inside both \( C_3 \) and \( C_4 \). Taking these two cycles and connecting them to make a curve, \( G \), as shown in \( d_{31} \), does not yield a diagram with abstraction \( \text{abstract}(d_{14}) + (G, \text{in}, \text{out}) \). This example illustrates the necessity of the constraint that any pair of cycles used to add a curve cannot have a common vertex inside them.

10 Conclusion

In the conclusion of the paper, we mentioned that curves can be removed whilst preserving all wellformedness conditions except, possibly, the connected zones property. Sometimes, when a zone becomes disconnected after curve removal, it is possible to ‘resolve’ this wellformedness condition by applying a transformation to the resulting diagram such that no other wellformedness conditions are broken and the zone is no longer disconnected.

Example 10.1: The diagram \( d_{32} \), Fig. 15, can be built using our inductive generation approach, starting with curve \( A \), then adding \( B \) and finally \( C \). With regard to wellformedness conditions, \( d_{32} \) possesses the simplicity property, no concurrency property, and connected zones property. It does not, however, possess the no triple points property or the crossings property; the point where \( A, B \) and \( C \) all intersect is both a triple point and a non-crossing point. Removing \( B \) gives \( d_{33} \) which, in addition to possessing the simplicity property and no concurrency property, also possesses the no triple points property. That is, removing \( B \) resolved the triple point. However, removing \( B \) also created a disconnected zone, that inside \( C \) only.

We can resolve the disconnected zone by pulling (at least one of) the (non-crossing) points where \( A \) and \( C \) intersect apart, as shown in \( d_{34} \). These points of intersection that are non-crossing points can be detected
Fig. 15. Removing curves and maintaining wellformedness.

by reading words around faces in the modified Euler dual (shown in $d_{35}$). Taking the Euler vertex, $v$, the word around the face in which $v$ is embedded is $c, a, a, c, \emptyset$. We can see from this word that $a$ and $c$ intersect at $v$, but do not cross. Moreover, we can see that an edge can be added to the modified Euler dual connecting the two vertices embedded in $c$ which passes through the vertex $v$ without crossing any edges. This indicates that we can pull $A$ and $C$ apart at $v$, resolving the disconnected zone and a non-crossing point; note that $d_{34}$ resolves both non-crossing points but, as demonstrated, this is not necessary in order to resolve the disconnected zone. Further, we observe that the manner in which we identified, and resolved, the non-crossing point could also be applied to $d_{32}$.

Currently, the authors are investigating diagram transformations that resolve broken wellformedness conditions.

**ACKNOWLEDGMENTS**

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

**REFERENCES**


Fig. 13. Ensuring drawability using many cycles.

Fig. 14. Cycles with a common vertex inside give the wrong abstraction.