Generating and Navigating Large Euler Diagrams

Aidan Delaney, Eric Kow, Peter Chapman and Jon Nicholson

University of Brighton,
United Kingdom
aidan@ontologyengineering.org
eric.kow@gmail.com
{p.b.chapman, j.nicholson}@brighton.ac.uk

Abstract. We automatically generate Euler diagrams that are too large to be readable on screen configurations commonly found in Universities and offices. We discuss how principles from information visualisation can be employed to render large diagrams intelligible. Furthermore, concrete examples implementing these information visualisation principles are presented. This work is at an early stage of development.

1 Introduction

Many visual logics are based on Euler diagrams, these include Peirce's α and β systems [10], first and second order spider diagrams [6, 3], constraint diagrams [7] and concept diagrams [17]. User studies involving Euler diagram based logics have considered their effectiveness for reasoning [14]. The effect of visual aspects of the Euler diagram token syntax has also been investigated [2]. Euler diagrams used in user studies typically contain fewer than 10 contours. Furthermore, visualisations using Euler diagrams, see [13] for an up-to-date survey, also tend to use a small number of contours.

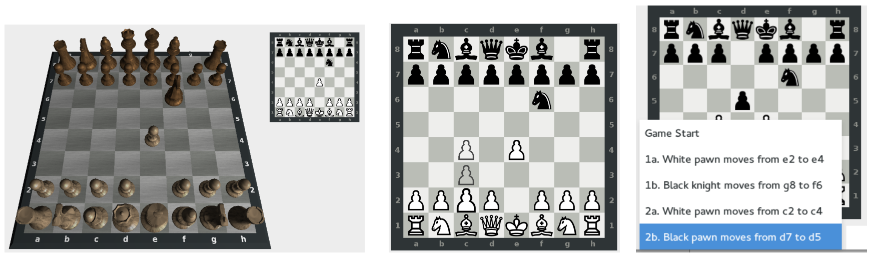
By contrast, we generate Euler diagrams with 100's of contours such that we may, in the future, establish the effectiveness of using large Euler diagrams for information visualisation. Such diagrams may arise when representing large inheritance hierarchies found in biological models, UML class diagrams and semantic web applications. Given our interest in these "real-world" application areas, we adopt a very lose definition of what is considered to be a large diagram. For our purposes, a large Euler diagram is one that is not readable when presented on a computer screen. In this paper we wish to provide a starting point to consider navigation of large Euler diagrams. In order to do this we automatically and pragmatically generate large Euler diagrams.

In investigating navigation in large Euler diagrams we are interested in mechanisms for implementing Shneiderman's information seeking mantra [15]. Within visualisation, Shneiderman's mantra considers providing:

– *overview* first,
– *zoom & filter*, then

– *details-on-demand.*

Example implementations of these principles can be seen in figures 1a, 1b and 6a respectively. The examples in figure 1 are drawn from the domain of computer games. In figure 1a a small scale map of the virtual world can be seen in the top right. It abstracts the graphical detail of the game in order to provide the player with an understanding of their situation within the game world. An implementation of a semantic zoom operation within the game, figure 1b, presents the possible area into which a particular game character (i.e. pawn $c2$) can move. Finally, details of the previous game moves can be accessed on demand, as seen in figure 6a. We now outline the structure of our discussion of how these operations can be applied to large Euler diagrams.


Game Start
1a. White pawn moves from e2 to e4
1b. Black knight moves from g8 to f6
2a. White pawn moves from c2 to c4
2b. Black pawn moves from d7 to d5

(a) (b) (c)

Fig. 1: Overview, zoom & filter and details-on-demand.

In section 2 we explain how we pragmatically generate large Euler diagrams. Following that, we provide examples of implementing overview functionality for large Euler diagrams; section 3.1. We proceed to address the zoom & filter concerns of Shneiderman's mantra in section 3.2. Thereafter, in section 3.3, we discuss the kinds of details-on-demand that are likely to be required in large Euler diagrams. Finally, we conclude with a discussion on how we expect to develop this work.

2 Generating Large Euler Diagrams

To generate large Euler diagrams, we first generate Euler diagrams containing a small number of contours. Following [11], we then use constraint based layout to compose the smaller diagrams into a large diagram. We make use of the separation between abstract syntax and concrete syntax to generate small Euler diagrams. A software tool, called `iCircles`, produces concrete layouts when given an abstract description. The `iCircles` tool is an implementation of the method, presented in [16], of inductively drawing Euler diagrams using circles. As such, the discussion that follows is restricted to consider Euler diagrams drawn only with circles and breaking a well-formedness property by allowing duplicate labels [12].

The abstract syntax of an Euler diagram is described by the tuple $\langle C, Z, Z^* \rangle$ where:

C is a finite set of *contours*,
Z is a subset of $\mathbb{P}C$ and denotes each *zone* of the diagram, and
Z^* is a subset of Z and denotes the *shaded zones* within a diagram.

Our presentation of a diagram zone differs from some of the literature on Euler diagrams. In the literature it is common to present a zone as a partition of C of the form (in, out) where $in \subseteq C$ and $out = C - in$. The above definition of zone matches the implementation within `iCircles`, where it is assumed that all contours in C appear in a concrete diagram.

An arbitrary Euler diagram is generated where $d = \langle C, Z, Z^* \rangle$ such that C is a set of contours. In theory the set of zones, Z, is an arbitrary subset of the set of all possible zones generated from C and Z^* is an arbitrary subset of Z. The inductive circles algorithm then produces a concrete layout from this abstract description. In practice, the `iCircles` implementation of the inductive circles algorithm contains faults that limit it to generating concrete layouts for abstract descriptions that describe *sparse* Euler diagrams. For n contours, the most sparse Euler diagram contains n contours that are pairwise non-overlapping, the least sparse diagram is Venn-n. Hence, we accede to pragmatism and take a heuristic approach when generating Euler diagrams. We can reliably generate arbitrary Euler diagrams with up to 6 contours. For higher numbers of contours we must enforce sparseness heuristics. Improving the implementation of `iCircles` remains a priority for future work.

Our heuristics for generating sparse Euler diagrams are implemented as a generator for arbitrary instances [4]. These arbitrary diagrams are consumed by `iCircles` and the concrete layout is produced as a scalable vector graphic [1]. Each diagram generated by `iCircles` is termed a “cluster” within the large Euler diagram. Constraint-based layout, provided by `WebCoLa`[1], is used to position these clusters within a larger diagram [11]. The constraint based layout ensures the individual clusters are placed close together but do not overlap. The example output in figure 2 was generated using this process. We will use this as a running example when discussing the aspects of Shneiderman’s mantra; overview, zoom & filter and details-on-demand. Using this approach, generating a large Euler diagram containing over 100 contours requires adding more small clusters to the large diagram.

3 Shneiderman’s mantra

Having seen examples of overview, zoom & filter and details-on-demand in figure 1 we now present an examples of how to implement each of these principles within large Euler diagrams.

[1] Available at https://github.com/tgdwyer/WebCola.

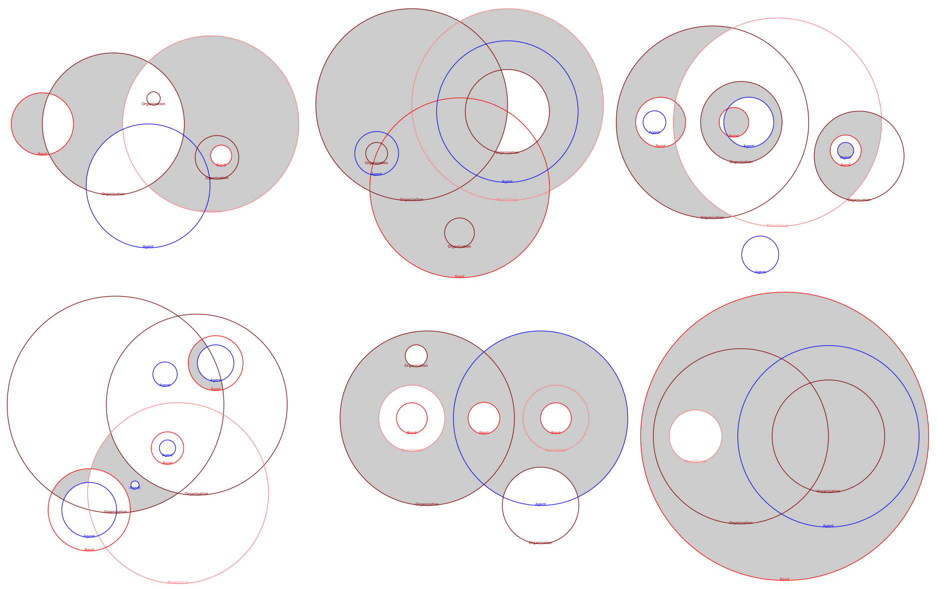


Fig. 2: An Euler diagram containing 51 circles.

3.1 Overview

The overview allows the user to orient themselves within the diagram. We have two mechanisms that support orientation within the diagram:

- a grid system acting in a manner similar to contour lines on a geographic map, and
- a smaller overview map embedded in the view.

Figure 3 depicts a grid super-imposed on the diagram. The super-imposed grid is rectilinear in order to contrast with the circular contours. Furthermore, the grid lines decrease in frequency with respect to the distance from the centre of the diagram. Using the grid, the operation of seeking a path to the centre becomes a local operation of using the next smallest grid cell as a waypoint. Our intention is that the grid provides some texture to the diagram. From all parts of the diagram the grid indicates the direction to the centre of the diagram. We have chosen the centre of the diagram as the focus of our navigation a the `WebCoLa` layout constraints specify an even distribution of the clusters around this position.

More conventionally, a small "map" of the entire diagram can be seen on the bottom left of figure 3. The map also displays an overlay of the current viewport. The user may use the small map to navigate towards a particular ma that is known to them. The small map may also allow a user to quickly jump to an area of interest without panning from their current location.

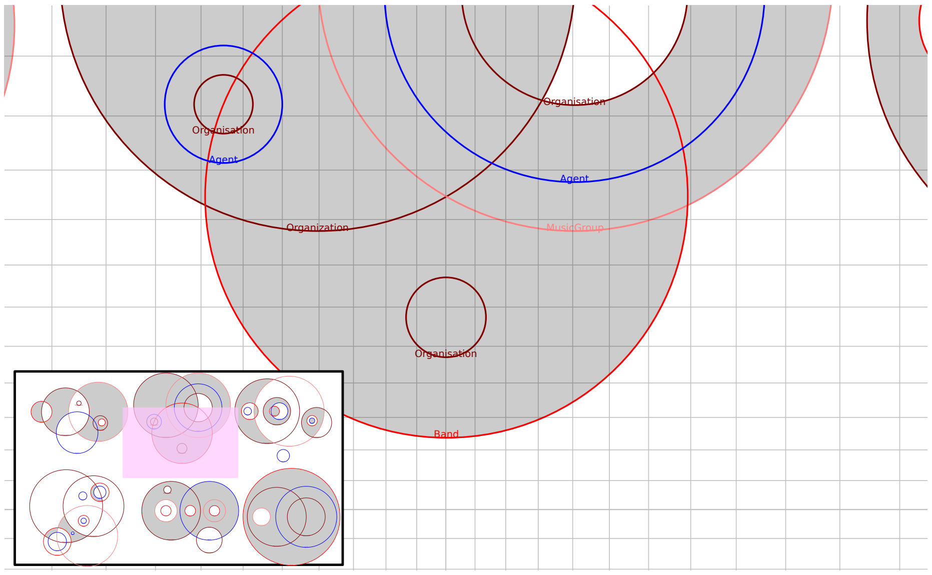


Fig. 3: The overview mechanism.

In both of our attempts to provide overview features for large Euler diagrams we adopt well-known techniques from cartography. From the abstract description of an Euler diagram, there is no way to reason about the concrete relative position of contours x and y in a large diagram. Therefore, it remains to be seen, through user studies, whether or not large Euler diagrams lend themselves to the cartographic navigational approach.

3.2 Zoom & Filter

Zooming also affects the small map. The viewport over the small map will increase in area as the user zooms out. Other concerns when considering zoom within large Euler diagrams include:

- the level at which to remove information from the visualisation, and
- the visual cost of not removing information from the visualisation.

To explore these issues, consider the deep containment hierarchy in figure 4a. In order to reduce visual clutter in the diagram it appears reasonable to hide a number of the innermost contours from the view. Their suppression can be indicated by employing ellipses. This raises the question of how often the most deeply nested contour within the hierarchy is considered, for practical purposes, to be more important than the intermediate levels? In such cases the highest and lowest level contours can be presented and the intermediate level contours are suitably elided. However, figure 4b presents an example where the intermediate levels of a deep containment hierarchy introduce three zones at the lowest levels.

Eliding the intermediate level contours in this example is not straightforward and suggests that there is no one-size-fits-all heuristic.

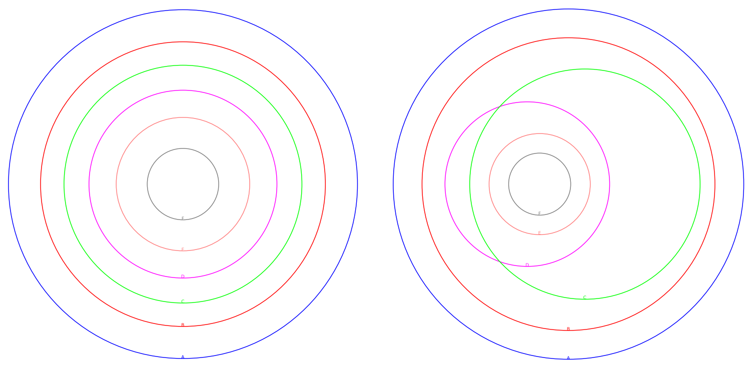

(a) (b)

Fig. 4: Deeply nested contours.

Rather than remove information from the diagram we view filtering as the process of improving the signal to noise ratio of the visualisation. Figure 5 a user searches for the name of a contour present in the diagram. Searching for a contour name highlights all of the occurrences of that contour within the diagram, both in the main view and within the small map. Within the main view, highlighting the contour is intended to boost its signal with respect to the background noise. Highlighting a contour within the small map is intended to aid navigation when panning across a large Euler diagram.

The interaction between zoom & filter within a large Euler diagram requires further exploration. Suppose a contour is only visible at the most detailed zoom level. Furthermore, the contour has been omitted from the current view in order to reduce the visual clutter. If a user then searches for the name of the hidden contour, should it be added to the current view and highlighted, or is the filter operation restricted to searching the current zoom level? Moreover, can a general tradeoff between zoom and filter be made for large Euler diagrams, or is the tradeoff dependant on the domain of application i.e. should the balance be different when visualising a software engineering class diagram as opposed to medical information?

3.3 Details on Demand

The principle of details-on-demand is less straightforward to apply to Euler diagrams. Euler diagrams represent sets and their relationships. There are no details other than those that are in plain view. However, it is still possible to

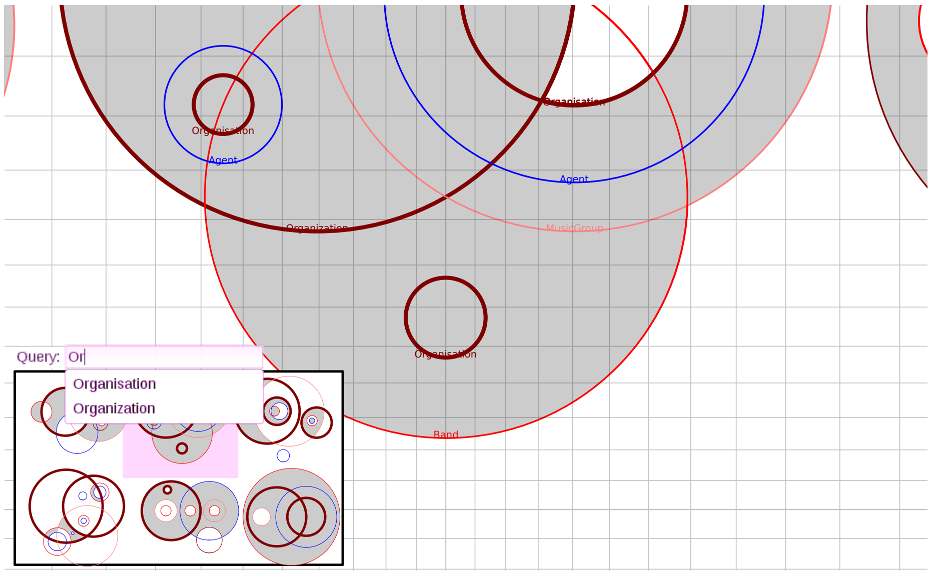


Fig. 5: The filter mechanism.

provide a user with details about a particular contour of interest. Figure 6a is an example depicting three clusters consisting of $2, 3$ and 4 contours. Suppose a user is interested the details of the highlighted contour. Figure 6b uses constraint based layout to vertically align clusters containing the interesting contour. In this configuration the user needs only scroll through the list of details, rather than navigate the entire diagram

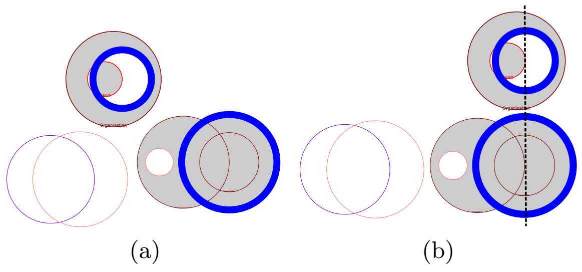


Fig. 6: A details on demand mechanism.

4 Conclusion

We have presented a pragmatic approach to generating large Euler diagrams for the purposes of exploring their navigation. The pragmatic approach combines existing tools to generate small Euler diagrams that are combined into large diagrams using constraint based layout. Furthermore, we have presented an outline implementation Shneiderman's information visualisation mantra within a viewer for large Euler diagrams.

The work is at an early stage of development. Our intention is to develop the Euler diagram generation such that it is using semantic web data from DBPedia [9]. Given real data and a usage context we will then run user studies to establish the tradeoffs involved in zoom & filter. Furthermore, given real data it will be possible to compare large Euler diagrams against visualisations such as topic maps [8] and conceptual graphs [5]. We are particularly interested in extending the inductive circles algorithm to layout concept diagrams [17].

References

1. http://www.w3.org/TR/SVG/ (2011)
2. Blake, A., Stapleton, G., Rodgers, P., Cheek, L., Howse, J.: The impact of shape on the perception of euler diagrams. In: Proceedings of the International Conference on the Theory and Application of Diagrams (2014)
3. Chapman, P., Stapleton, G., Delaney, A.: On the expressiveness of second-order spider diagrams. Journal of Visual Languages & Computing 24(5), 327 – 349 (2013)
4. Claessen, K., Hughes, J.: Testing monadic code with QuickCheck. SIGPLAN Notices 37(12), 47–59 (Dec 2002), http://doi.acm.org/10.1145/636517.636527
5. Dau, F.: The Logic System of Concept Graphs with Negations: And its Relationship to Prediacte Logic. Springer Verlag (2003)
6. Howse, J., Stapleton, G., Taylor., J.: Spider diagrams. LMS Journal of Computation and Mathematics 8, 145–194 (2005)
7. Kent, S.: Constraint diagrams: Visualizing invariants in object oriented modelling. In: International Conference on Object Oriented Programming, Systems, Languages and Applications. pp. 327–341. ACM Press (October 1997)
8. Le Grand, B., Soto, M.: Visualisation of the semantic web: Topic maps visualisation. In: Information Visualisation, 2002. Proceedings. Sixth International Conference on. pp. 344–349 (2002)
9. Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P.N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., Bizer, C.: DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. Semantic Web Journal (2014)
10. Peirce, C.: Collected Papers, vol. 4. Harvard University Press (1933)
11. Riche, N.H., Dwyer, T.: Untangling Euler diagrams. Visualization and Computer Graphics, IEEE Transactions on 16(6), 1090–1099 (2010)
12. Rodgers, P., Zhang, L., Stapleton, G., Fish, A.: Embedding wellformed Euler diagrams. In: 12th International Conference on Information Visualization. pp. 585–593. IEEE (2008)
13. Rodgers, P.: A survey of euler diagrams. Journal of Visual Languages and Computing 25, 134–155 (2014)

14. Sato, Y., Mineshima, K., Takemura, R.: The efficacy of Euler and Venn diagrams in deductive reasoning: Empirical findings. In: Proceedings of the International Conference on the Theory and Application of Diagrams. pp. 6–22 (2010)
15. Shneiderman, B.: The eyes have it: a task by data type taxonomy for information visualizations. In: Visual Languages, 1996. Proceedings., IEEE Symposium on. pp. 336–343 (Sep 1996)
16. Stapleton, G., Zhang, L., Howse, J., Rodgers, P.: Drawing euler diagrams with circles: The theory of piercings. IEEE Transactions on Visualization and Computer Graphics 17(7), 1020–1032 (July 2011)
17. Stapleton, G., Howse, J., Chapman, P., Delaney, A., Burton, J., Oliver, I.: Formalizing concept diagrams. In: 19th International Conference on Distributed Multimedia Systems. pp. 182 – 187. Knowledge Systems Institute (2013)