Layout Decisions for Tabular Euler Diagrams

Uta Priss

Fakultät Informatik, Ostfalia University, Wolfenbüttel, Germany
www.upriss.org.uk

Abstract. Euler diagrams as a means for visualising sets and their intersections tend to be easy to learn to read but difficult or even impossible to draw. Algorithms for automated drawing exist but do not necessarily produce optimal layouts. This short paper describes a method for creating a well-readable layout for certain types of Euler diagrams. The method consists of translating an Euler diagram into an *Attribute Relationship Graph (ARG)* which is defined in this paper. An ARG is then evaluated by applying a set of rules using standard graph drawing software. This semi-automated method produces better layouts than automated layouts and is much faster than complete manual construction.

1 Introduction

Euler diagrams are a well-known means for visualising sets and their intersections or, in other words, for partially ordered sets. Formal Concept Analysis (FCA) is a technique for generating partially ordered sets (or more precisely 'concept lattices') from data (Ganter and Wille 1999). Traditionally, FCA employs Hasse[1] diagrams for visualisations but using Euler diagrams instead has the advantage of requiring less user training (Priss 2017). A disadvantage of Euler diagrams is that they are much more difficult to automatically generate and that in some cases workarounds (such as 'splitting attributes') are required in order for a diagram to be drawable at all. Software for automated drawing of Euler diagrams exists but tends to not emphasise well-readable layouts[2] because just generating diagrams is already challenging.

We have previously suggested that Euler diagrams in the shape of tables, called *tabular diagrams*, are preferable (Priss 2021). In our experience, it can take hours of trial and error to draw a tabular diagram by hand without tools because depending on its layout, a diagram can be visually appealing or difficult to read. FCA software[3] exists that lets users manipulate Hasse diagrams by clicking on nodes and dragging them around. Similar software for tabular diagrams would be very challenging to implement because there are more restrictions for what is allowed for Euler diagrams. Furthermore, if a diagram is not drawable without a workaround then the diagram could not be displayed at all unless some workaround had been preselected which would limit the users' choices

[1] The diagrams should not be named after Hasse because he did not invent them, but the name is widely established in the literature.

[2] For example, Paetzold et al. (2023) add unnecessary zones; Rodgers (2014) mentions several software tools that mostly produce irregular, concave shapes.

[3] https://upriss.github.io/fca/fcasoftware.html

on how to proceed. Therefore, Euler diagram drawing algorithms often consider not the diagram itself but a graph (in the sense of mathematical graph theory) that shows adjacency. The method suggested in this paper also employs a mathematical graph but it is a somewhat different construction and thus a novel approach.

The idea is to generate an *Attribute Relationship Graph (ARG)*, which focuses on some core features of the data. Because an ARG is a mathematical graph, it can be imported into existing graph drawing software where its vertices can be moved around until some rules are met. If the rules cannot be met then it is quite likely that a tabular diagram without workarounds is not drawable at all. In the future, software could be implemented that automates the rules. But manipulating an ARG by hand with existing graph drawing software already significantly reduces the amount of time required to create an appealing layout for a tabular diagram as demonstrated by the examples in this paper. Several generalisations of the approach suggested in this paper are possible: applications to other types of diagrams but also extending the use of ARGs by incorporating further types of relationships.

Because this paper continues our previous research of representing concept lattices with Euler diagrams not everything that was defined by Priss and Dürrschnabel (2024) is repeated. But the main notions are explained using examples. Similarly, the definitions of FCA are not repeated in this paper but can be found in the book by Ganter and Wille (1999). Although some FCA terminology is used in this paper, for example, the data is collected in *formal contexts* and the sets of an Euler diagram are called *attributes*, this paper should be mostly comprehensible without any FCA knowledge. The following section defines tabular Euler diagrams. Section 3 describes the method for constructing Euler diagrams as the main contribution of this paper. The paper finishes with a concluding section.

2 Tabular Euler Diagrams

Euler diagrams are a means for visualising sets and their partial orders where attributes (or sets) are represented by closed curves with objects (or elements) enclosed by the curves. Priss (2021) argues that *tabular diagrams* are preferable. In a rectangular Euler diagram (or *rectangular diagram*) each set is represented by a rectangle – sometimes with rounded corners for better discernibility. A tabular diagram is a rectangular diagram where all rectangles are either adjacent sets of rows or of columns of a table. A one-dimensional tabular diagram consisting of a single row or single column is called a *linear diagram*. In order to draw a tabular diagram for a given formal context, the set of attributes needs to be divided into two partitions so that each partition is drawable as a linear diagram (one for the rows and one for the columns). If that is impossible then a tabular diagram is not drawable. Dürrschnabel and Priss (2024) discuss criteria for deciding drawability. More details about and an overview of Euler diagrams in general are provided by Rodgers (2014) and Alsallakh et al. (2016).

The main notions of Euler diagrams can be illustrated using an example of a lattice of types of triangles (cf. Ganter and Wille 1999) and its modelling as a tabular diagram in Fig. 1. Rectangles that are contained in each other indicate a subset relationship. In Fig. 1 the rectangle for ‘oblique’ includes the rectangles for ‘acute’ and ‘obtuse’.

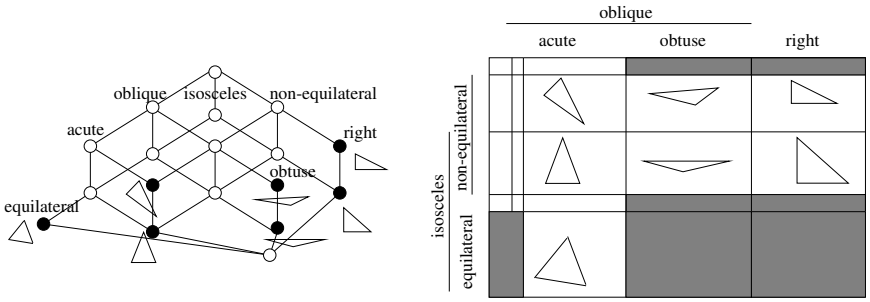

Fig. 1. A lattice for types of triangles and its corresponding tabular diagram

Thus the set of acute objects is a subset of the set of oblique objects and the attribute 'acute' implies the attribute 'oblique'. The cells of the table are called *zones*. Each zone either contains an object (eg. the triangles in Fig. 1) or is empty (corresponding to a concept without immediate objects) or is shaded (not corresponding to a formal concept). A well-formedness condition for tabular diagrams in this paper is a one-to-one correspondence between non-shaded zones and concepts of an FCA lattice apart from the bottom concept which can be omitted. In Fig. 1, the top concept is the small zone in the top left corner, the bottom concept is omitted. In other applications the top concept can also be the zone outside of the table. Euler diagrams are more general than Venn diagrams which have as many zones as a complete powerset.

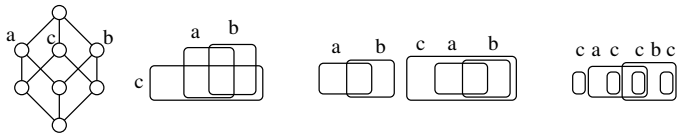

Fig. 2. Hasse diagram and tabular diagrams with different layouts

Unfortunately, not all partial orders are drawable as tabular diagrams. For example, a Boolean lattice with 4 attributes is drawable as a tabular diagram but not with 5 attributes unless some workarounds are applied, such as splitting attributes. Figure 2 displays a Boolean lattice with 3 attributes as a Hasse diagram, as a tabular diagram with 4 columns and two linear diagrams with split attributes. All four diagrams are *zone-equivalent* because they represent the same set of zones. In these tabular diagrams, attribute c is treated visually differently from a and b which should ideally correspond to a semantic difference as well. Thus the meaning of attributes can be relevant for deciding which tabular diagram is most appropriate and certainly more important for the layout of Euler diagrams than it is for Hasse diagrams.

3 Layered Attribute Relationship Graphs

Dürrschnabel and Priss (2024) provide criteria for deciding whether or not a lattice is drawable as a tabular diagram. But even if a lattice is drawable it can still be a challenging task to determine a well-readable layout. Therefore a method for deciding how to partition and arrange the attributes as presented in this section is helpful. The method relies on investigating the relationship between the extensions for all pairs of attributes. The following definitions assume that a and b are attributes with non-empty extensions[4].

a and b are ...	if this is true for their extensions:	this implies ...
comparable	$a' \subseteq b'$ or $b' \subseteq a'$	not contrary, not co-related
contrary	$a' \cap b' = \emptyset$	not comparable, not co-related
co-related	$a' \cap b' \neq \emptyset$ and $a' \not\subseteq b'$ and $b' \not\subseteq a'$	not contrary, not comparable
equal	$a' = b'$	comparable
dichotomic	$\forall o$: either $o \in a'$ or $o \in b'$	contrary

The property 'equal' is a special case of 'comparable', and 'dichotomic' is a special case of 'contrary'. For equal attributes, one can be omitted from the data. An attribute with an empty extension must either be omitted or some 'placeholder' object must be added to its extension. Evaluating properties for all pairs of attributes does not significantly increase the computational complexity because there are only $n(n-1)/2$ pairs for n attributes. Furthermore, FCA software computes extensions and visualisations are usually only of interest for smallish numbers of attributes (e.g. less than 20) anyway.

Lemma 1. *Two attributes a and b (with non-empty extensions) are always either* comparable *or* contrary *or* co-related*.*

Proof. Further logical combinations of the properties yield nothing different. For example, ($a' \cap b' \neq \emptyset$ and ($a' \subseteq b'$ or $b' \subseteq a'$)) is equivalent to ($a' \subseteq b'$ or $b' \subseteq a'$).

It would also be possible to consider relationships between triples or larger sets of attributes but it is not clear whether the additional effort provides much additional benefit. One can define a set of attributes as *independent* if every possible conjunction occurs which means that the attributes form a Boolean lattice. Instead of 'dichotomic', similar properties 'trichotomic' and so on can be defined. The first three properties are collected in an *Attribute Relationship Graph (ARG)* which has attributes as vertices and an edge for each pair that is comparable or co-related. Furthermore, the edges for comparable attributes are drawn as dashed arrows (pointing to the superset) and the edges for co-related attributes as normal lines. In addition, a list of dichotomic pairs can be collected (notated as $a <> b$). Figure 3 shows the attribute relationship graph for the example in Fig. 1. The layers are explained below. Vertices that are not connected by an edge are contrary. The ARG supports layout decisions as explained below and also often determines whether a tabular diagram is drawable at all.

It is known from FCA that if one divides a set of attributes into two partitions, generates a lattice for each partition and creates the direct product of these two lattices,

[4] Using FCA terminology: an attribute (or 'set label') a defines an extension a' which is a set of its elements (or objects.)

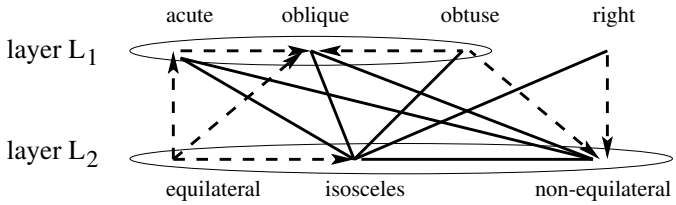

Fig. 3. Attribute Relationship Graph

then the original lattice can be embedded into the product lattice (Theorem 7 of Ganter and Wille (1999)). Traditionally, this principle is used in FCA for creating nested line diagrams. It also applies to tabular diagrams if each of two lattices is drawable as a linear diagram.

Lemma 2. *A tabular diagram of a concept lattice is drawable if and only if the set of attributes of a formal context can be divided into two partitions so that the lattice of each partition is drawable as a linear diagram.*

Proof. Each zone of a table belongs to exactly one row and one column. The index of each zone corresponds to a pair of column and row index. Thus the zones correspond to a direct (or Cartesian) product of rows and columns which is exactly the same situation as in Theorem 7 of Ganter and Wille (1999).

A *layered attribute relationship graph (or LARG)* is an ARG where the vertices are arranged in two horizontal layers[5] (L_1 and L_2) corresponding to the two partitions of the set of attributes as in Fig. 3. The following rules for checking a LARG support layout decisions as explained below. The rules should be processed in the provided order, jumping back from R6 to R1 if a resulting diagram is not drawable until the diagram is drawable or no further possibilites of partitioning the attributes are feasible.

R1 Both layers should contain roughly the same number of vertices.
R2 Within L_1 and L_2 only neighbouring vertices can have a solid edge between them. (If not possible, go back to R1 or STOP)
R3 Within L_1 and L_2 there should be as few solid edges as possible.
R4 Dashed edges should be contained within one layer but edge crossings should be minimised.
R5 If it is not possible to have all dashed edges from one vertex in one layer, then it may be better to minimise the number of dashed edges within each layer.
R6 Dichotomic attributes should be in the same layer.
R7 The meaning of the attributes can be considered for sorting them within the layers.

The reason for R2 is that if the extensions of two attributes have a non-empty intersection, a zone for this intersection needs to be placed between the attributes. If three

[5] An alternative but more traditional graph theoretical modelling might use edge colouring instead of layers: one colour for the edges amongst vertices in L_1, another colour for edges within L_2 and a third colour for the remaining edges.

attributes are mutually co-related but not comparable then a linear diagram is only drawable if the extension of one of the attributes is contained in the union of the extensions of the others which means some potential zones are missing. In some rare cases missing zones are not problematic (as discussed by Priss and Dürrschnabel (2024)). But for applications, the probability is extremely high that a tabular diagram is not drawable if R2 cannot be fulfilled. Thus, even though R2 is not strictly necessary from a mathematical view, it can be treated as a necessary condition in applications. Furthermore the readability of a tabular diagram that contradicts R2 is likely to be poor.

The other six rules are not necessary but impact the readability of tabular diagrams and are therefore crucial for determining a layout. Quite often they cannot all be fulfilled simultaneously and need to be balanced against each other. Rules R1 and R3 both make it more likely that linear diagrams are drawable and reduce the numbers of zones of the table in total which implies fewer shaded zones. Rule R4 also reduces shading and improves readability. For example, R4 is contradicted in Fig. 3 by several attributes and as a consequence, it is much easier in Fig. 1 to see that 'acute' and 'obtuse' are comparable with 'oblique' than it is to see that 'right' and 'obtuse' are comparable with 'non-equilateral'. R5, R6 and R7 are the weakest rules and should only be applied if there is anything left to decide. R3 already implies that contrary attributes tend to be in the same layer because there is no edge between them. R6 is therefore a special case of R3.

Figure 4 shows LARGs that were drawn following the rules. In the first example, R2 determines that the 4 attributes in the middle must be distributed across the 2 layers. Initially any of the 4 remaining attributes could be in the same layer. R3 determines that 'running' and 'artificial' should be in different layers. R5 then determines that 'stagnant' and 'natural' should also be in different layers and the locations of 'temporary' and 'maritime'. The attributes 'constant' and 'inland' could be switched which would also switch 'temporary' and 'maritime' but the resulting tabular diagram would have a similar layout that is neither better nor worse. In the second example, R4 is the driving force. The ordering of the first and last two attributes in each layer is determined by R7. Both LARGs correspond to tabular diagrams presented by Priss (2021) which confirms that these previously manually designed diagrams were quite likely optimal with respect to reducing the number of shaded zones and good readability apart from trivial changes such as mirroring the diagrams or moving attributes that do not impact the overall layout structure.

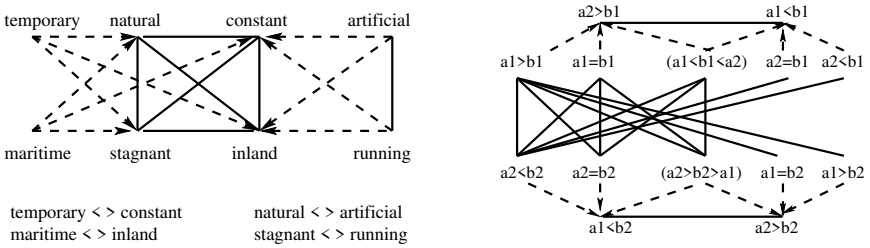

Fig. 4. LARGs that follow the rules

Figure 5 shows an example from Ganter[6] (2013, p. 76) which contains a Boolean lattice with 4 attributes (f inj, g inj, f surj and g surj). The LARG on the right side is determined by R2 and R4 and results in the tabular diagram shown in Fig. 6. Thus while there may be many choices for the layout of the Hasse diagram in this example, the layout of the tabular diagram is fairly deterministic. In our opinion, the tabular diagram shows the underlying Boolean lattice with 4 attributes more clearly than the Hasse diagram.

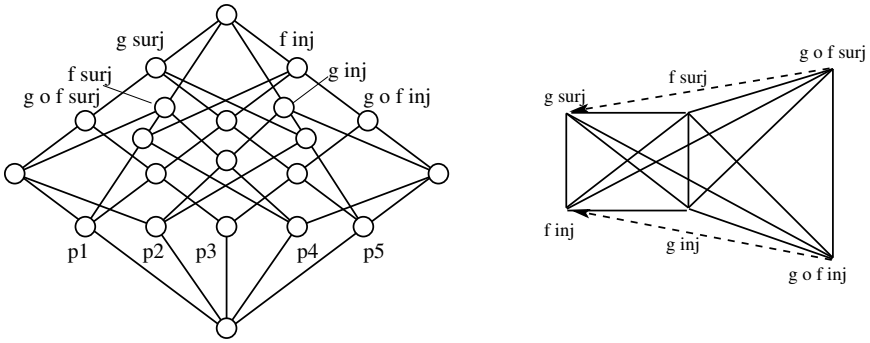

Fig. 5. Hasse diagram and LARG for an example from Ganter (2013, p. 76)

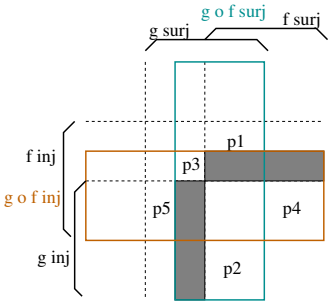

Fig. 6. Tabular diagram derived from the LARG in Fig. 5

4 Conclusion

In summary, this paper continues previous research on tabular diagrams by providing a semi-automated method for deriving layouts that minimise shaded zones and support vi-

[6] This example was chosen because Bernhard Ganter mentioned to me that this was maybe the lattice which he found most difficult to draw with a layout that he liked.

sual recognition of attribute implications. It is interesting to note that all of the 'famous' lattice[7] examples from Ganter and Wille (1999) and Ganter (2013) that we have looked at so far appear to be drawable as tabular diagrams even though many random concept lattices are not drawable. It is possible that whatever made these examples appealing to Ganter and Wille also makes them more likely to be drawable as tabular diagrams. The reasons are probably symmetry and sufficient attributes that are comparable or contrary to each other.

For future research it would be of interest to explore whether there are further applications for ARGs. An advantage of ARGs is that properties can either be derived from a formal context or, in case of incomplete data, asserted for the data based on background knowledge.

References

1. Alsallakh, B., Micallef, L., Aigner, W., Hauser, H., Miksch, S., Rodgers, P. (2016). The State-of-the-Art of Set Visualization. Computer Graphics Forum, 35, 1. p. 234-260.
2. Dürrschnabel, D., Priss, U. (2024). Realizability of Rectangular Euler Diagrams. In: Cabrera et al. Conceptual Knowledge Structures. Proceedings of Concepts 2024, Springer Verlag, LNAI 14914, 2024, p. 149-165.
3. Ganter, B.; Wille, R. (1999). Formal Concept Analysis. Mathematical Foundations. Berlin-Heidelberg-New York: Springer.
4. Ganter, B. (2013). Diskrete Mathematik: Geordnete Mengen. Berlin-Heidelberg-New York: Springer.
5. Paetzold, P., Kehlbeck, R., Strobelt, H., Xue, Y., Storandt, S., Deussen, O. (2023). Recteuler: Visualizing intersecting sets using rectangles. Computer Graphics Forum 42.
6. Priss, U. (2017). Learning Thresholds in Formal Concept Analysis. In: Bertet; et al. (eds.), Formal Concept Analysis. Proceedings of ICFCA'2017, Springer Verlag, LNCS 10308, p. 198-210.
7. Priss, U. (2021). Visualising Lattices with Tabular Diagrams. In: Proceedings of DIAGRAMS'21, LNAI 12909, Springer.
8. Priss, U., Dürrschnabel, D. (2024). Rectangular Euler Diagrams and Order Theory In: Proceedings of Diagrams 2024, Springer Verlag, LNAI 14981, 2024, p. 165-181.
9. Rodgers, P. (2014). A survey of Euler diagrams. Journal of Visual Languages & Computing, 25, 3, p. 134-155.

[7] The lattice for the left LARG of Fig. 4 is on the title page of Ganter and Wille (1999).