Recognizing Sketches of Euler Diagrams Drawn with Ellipses

Aidan Delaney[1] Beryl Plimmer[2] Gem Stapleton[1] Peter Rodgers[3]
[1]Visual Modelling Group, University of Brighton, UK.
{a.j.delaney,g.e.stapleton}@brighton.ac.uk
[2]University of Auckland, New Zealand. beryl@cs.auckland.ac.nz
[3]University of Kent, UK. p.j.rodgers@kent.ac.uk

Abstract

Euler diagrams form the basis of a number of visual languages. However, the existing tool support for creating Euler diagrams is limited to generic diagram editing software that uses mouse and keyboard interfaces. A more natural and convenient mode of entry is via a sketching interface. In addition, it is known that sketching, as opposed to using an editing package, facilitates greater cognitive focus on the task of diagram creation. This paper presents the first sketch tool for Euler diagrams. In particular, we describe the recognition mechanism and conversion that takes a sketched Euler diagram and converts it into a formal diagram drawn with ellipses and circles. The semantics of the sketch is computed.

1. Introduction

Euler diagrams are a popular and widely used tool for information visualization. In addition, they form a component of many visual languages, such as spider diagrams [8], Euler/Venn diagrams [21], Venn-II diagrams [19], constraint diagrams [12], and ontology diagrams [15]. Given their wide-ranging practical use, there is a need to provide convenient ways of creating these diagrams in electronic form. Moreover, the software in which they are created will, ideally, have some understanding of the diagram semantics, so it can further support the user in exploring information conveyed by the diagram. Currently, however, the manual creation of Euler diagrams in electronic form relies on unintuitive mouse and keyboard interfaces in software that has no semantic understanding of the diagram.

A natural creation method for Euler diagrams is using a pen, but no intelligent tool support exists for this mode of entry. The lack of sketching support for Euler diagrams means that, in the vast majority of cases, when using a computer they must be drawn in off-the-shelf diagram editing tools. The current editing support does not provide a natural and convenient interface because the point-by-point specification of the diagrams' curves is slow (compared to sketching) and the act of sketching is simple in comparison to using a diagram editor, such as those found in Microsoft's Word or Visio packages. Sketching the diagram has the further advantage that it allows the user to focus on the actual diagram creation rather than the interface of the editing tools.

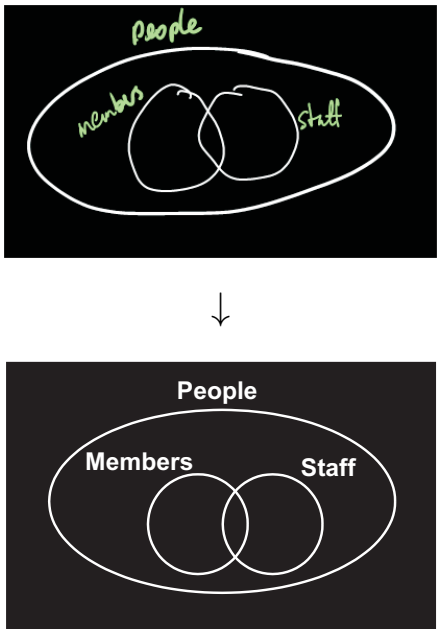

Figure 1. A sketched Euler diagram and its formal visualization.

The sketch recognition software developed to date has focused on user interface design and graph oriented diagrams [11]. With respect to user interface design tools the sketched items are largely independent of each other. In graph oriented diagrams the spatial positioning of nodes and edges are not of semantic significance. By contrast, in Euler

diagrams the spatial relationships between sketched items is fundamental to their semantics. To our knowledge, the work described in this paper is the first to consider these types of complex spatial relationship for sketch recognition.

This paper describes a mechanism for recognizing sketches of Euler diagrams drawn with ellipses. This work forms a basis for developing more sophisticated sketch recognition tools for Euler diagrams and, hence, for the many notations that extend them. To illustrate the process, a user-sketched Euler diagram can be seen in the top of figure 1 whereas a diagram drawn in an editing tool is placed underneath; we call this lower diagram a *formal* diagram. The sketch tool that we have developed can automatically produce a formal diagram from the user sketch. Our mechanism relies on, firstly, dividing the syntax into two classes: handwritten labels and hand-drawn curves. The curves are further classified as ellipses or circles. Syntax identified as a label is subsequently passed to a standard handwriting recognition package. Syntax identified as a curve is converted into a formal representation by finding a suitable approximating circle or ellipse, as appropriate.

Section 2 provides background material on Euler diagrams and sketch recognition, thus providing motivation for the research and contextual information. Section 3 describes our Euler diagram sketch recognition techniques, including our method for deriving the diagram semantics. Finally, we conclude in section 4, where we also discuss further directions for this research.

2. Context and Motivation

Euler diagrams [4] are collections of closed curves used to visualize relationships between sets and generalize Venn diagrams [23]. Euler diagrams are a common component of visual languages because they intuitively represent exclusion, containment and intersection of sets. An example can be seen in figure 2, which shows a categorization of countries in the British Isles. The varied application areas of Euler diagrams include, but are not limited to: crime control [5], computer file organization [3], classification systems [22], education [10], genetics [13], and medicine [20]. Thus, there is widespread interest in these types of diagrams, but under-developed tools to support their creation. Providing sketch tools for Euler diagrams has the potential to be beneficial to a large community of users.

Sketching a diagram on a piece of paper is a useful problem solving and communications technique [7]. In a sketching context, users can produce, evaluate, modify, refine and replace diagram components rapidly. The backtalk from the external representation of the diagram is considered an essential part of the cognitive support for design. If a computer 'understands' the sketch it can be more easily edited, animated and translated from one format to another. Pen-input computers mean that the act of drawing can now be performed directly on the computer screen or tablet. Automatic and reliable recognition of hand-drawn diagrams is now becoming possible.

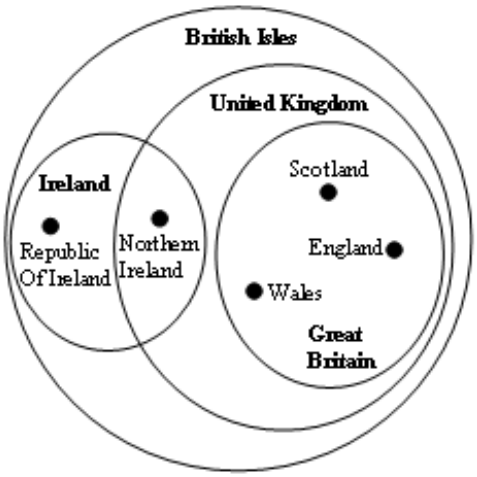

Figure 2. The British Isles, by Sam Hughes [9].

Hand-drawing diagrams has been shown to be more effective for external representation of a problem than using formal computer diagramming tools [6]. However to retain these benefits the computer-based sketch tools interaction must be carefully designed with particular attention to retaining the hand-drawn appearance of the diagram [18] in the initial stages of the diagram's creation. Underlying such a sketching interface there must be a sophisticated and accurate recognition engine. While much of the early work was on recognition engines for specific diagrams, for example [14], there are now a number of configurable or trainable recognition engines [16, 17].

However, there are no recognition engines specifically designed for diagrams where the spatial relationship between the syntactic components is of primary significance. For instance, closed curves (typically circles) represent nodes in graph-like diagrams and their spatial relationship is often taken to have no semantic relevance whereas closed curves in Euler diagrams overlap or enclose one another precisely to convey semantically relevant information: overlapping and containment between curves is unique and critical to Euler diagrams.

3. Recognizing Euler Diagrams

There are various stages to our recognition process which we have implemented in a prototype software tool.

This tool is an extension of the existing SketchNode codebase [18]. Given a user created sketch, the first stage is to divide the sketched components into two categories: labels and curves. We derive the semantics of the sketch by computing its abstract syntax. Independently, we convert the sketched curves into formal curves; in this paper we restrict the curves to being to ellipses, with circles as a special case of an ellipse. The curve labels are processed by a handwriting recognition engine.

3.1 Dividing the Syntax

An example of a sketched Euler diagram, drawn in our software tool, can be seen in figure 3. The first phase of the recognition process is to classify each syntactic component as a curve label or as a curve; curves are further categorized as circles or ellipses. We use AI techniques to derive

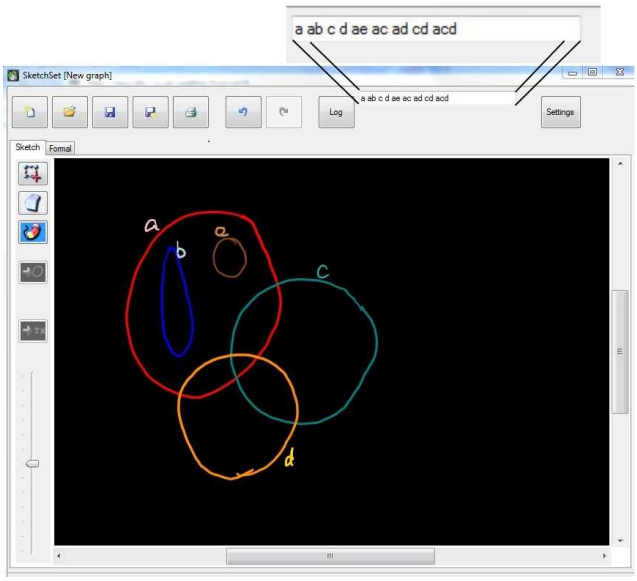

Figure 3. A sketched Euler diagram.

a mechanism for the classification. The recognizer component, Rata.SSR [2], was trained from a number of sample diagrams; we chose 10 Euler diagrams, each with up to 4 curves. The curve labels comprised the letters of the alphabet. We asked 10 participants to sketch each of the 10 Euler diagrams and label their curves appropriately. This provided us with a training set of 100 sketches. The sample diagrams were collected in DataCollector, part of DataManager [1].

To illustrate, figure 4 shows a screenshot of a participant's sketch. The righthand panel shows the description of the task. Here, two descriptions of Euler diagrams are given and the participant is asked to draw them. The top sketch was drawn from the Euler diagram description f, fg, fh which identifies the regions to be present in the diagram; f is a region inside just a curve labelled f, fg is inside just curves labelled f and g and so forth. The participants who were not familiar with this manner of describing an Euler diagram were provided with training.

After collecting the 100 sketches, we labelled each digital ink stroke as: text, curve-circle or curve-ellipse. From the sketches, DataManager generates feature vectors from each ink stroke and uses these to train the Rata.SSR generator. The Rata.SSR is then consumed within our software tool as a component. It separates labels from curves and recognizes curves as circles or ellipses. Labels are passed to the operating system text recognizer while the curves are used to generate the equivalent formal visualization; we discuss this latter aspect in section 3.3.

3.2 Abstract Syntax of the Sketch

A sketch tool needs semantic understanding of the diagram in order to support intelligent interaction such as editing and to generate other representations. Euler diagrams represent sets using curves and the diagram's *zones* completely determine the relationships between the sets. That is, the zones present correspond to the semantics. A zone is a region that can be described as being inside some curves but outside the rest of the curves. For example, the sketch in figure 3 has zones described by a, ab, c, d, ae, ac, ad, cd, acd, as shown in the callout. This list of zone descriptors is called the *abstract description*.

In order to compute the abstract description, we first assign each curve label its closest curve, unless that curve is already assigned a label. The process of label assignment is done on-line, whilst the user is creating the sketch, and adjustments are made as the user performs edits to the diagram. For instance, if a curve is deleted but its associate label remains then that label may be reassigned to the next closest curve, or assigned to no curve if every sketched curve already has a label. Each curve is automatically assigned a colour that is used in both the sketch and formal views. There is an internal list of 8 colours for curves; the colours are automatically assigned in sequence as curves are recognised. Curve labels are assigned a lighter shaded of the colour associated to their curve. This allows the user to readily check the association made by the software and, thus, have an opportunity to change the sketch if necessary.

There are three major steps when building the abstract description of a sketched Euler diagram: first, all the separate zones in the diagram must be identified; second, we establish the containment relationships between zones; lastly, we build the abstract description by iterating through all zones to establish their label.

The zones are created by first identifying intersections between each pair of (real) curves, c_1 and c_2. When an in-

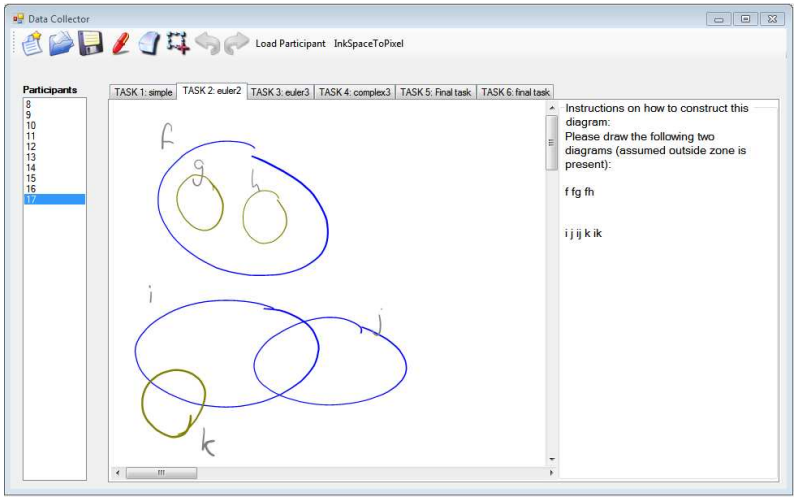

Figure 4. A sketch from the training set.

tersection is found between c_1 and c_2, a virtual curve, c_3, is created with a sequence of points made from the segments of the real curves that form the boundary of the intersection space. Information on the parent curves, c_1 and c_2, is held in the virtual curve, c_3, for later use. In figure 3, this results in virtual curves ac, ad and cd (three regions formed from the pairwise intersecting curves). The algorithm then checks the set of virtual curves looking for intersections. On the next pass, the zone acd is discovered, for example. If more than one new virtual curve is created the algorithm calls itself with the new virtual curves looking for further intersections.

We also need to identify when one curve, virtual or real, contains another. To establish containment, the smallest (if any) container for each curve is found. Each curve (real and virtual) is checked against all other non-parent curves (a virtual curve will always fall within its parents' bounding boxes). When the bounding box of a curve lies within the bounding box of another curve then the other curve is a candidate container. If more than one candidate container is found then the smallest is the container the outer containers can be found by check each containers container.

The abstract description can then generated by building the label for each curve. A label for a real curve consists of the label of its container (which may itself have a container) and its own label. The label for a virtual curve is the set of the labels of the component curves To allow the abstract description to be constructed when a curve is unlabelled the curve's colour is used as shown in the callout in figure 5.

3.3 Converting to the Formal Diagram

We now describe the process by which we convert a sketch drawn curve into a formal curve. First, we describe

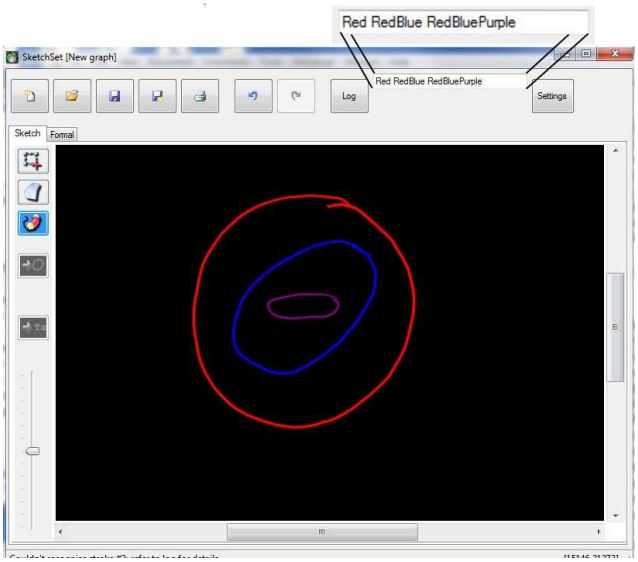

Figure 5. A sketch with no curve labels.

how to convert a sketched curve that has been classified as an ellipse, but not a circle, into a formal ellipse. A sketched curve, see the left of figure 6, is represented internally by a sequence of points, as shown on the right. We find the longest chord across the ellipse by taking the two points that are furthest apart in the internal representation and we then draw a line segment between them. We can calculate the rotation of the sketched ellipse from the angle, α, formed by this chord and the x-axis, as shown in figure 7.

Next, we rotate the sketched ellipse by α so that this chord is parallel to the x-axis. A standard algorithm is then used to find a bounding box of the sketched ellipse, as shown in figure 8. This rectangle is used to create the

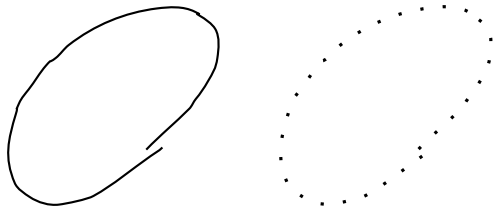

Figure 6. A sketched ellipse and its internal representation.

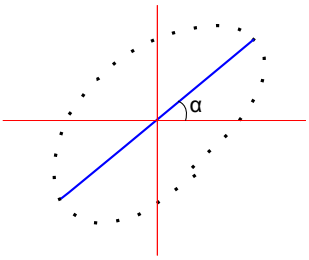

Figure 7. Computing the angle of rotation.

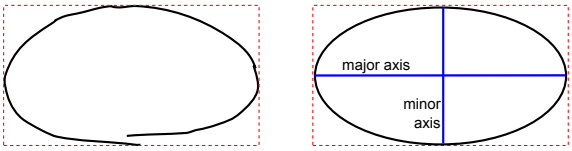

Figure 8. Finding a bounding box to compute the minor axis.

formal ellipse: the centre of the ellipse is the centre of the rectangle, the major axis takes the width of the rectangle, and the minor axis is the height of the rectangle. Finally, the formal ellipse has rotation α, the same angle of rotation as the sketched version, shown in figure 7.

However, we attempt to preserve user intent in this last step, with regard to approximate alignment with the x and y-axis. If α is within 10 degrees of 0, 90, 180, or 270 degrees then we round α accordingly to achieve alignment.

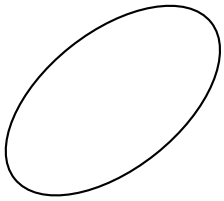

Figure 9. The final formal ellipse.

In the case of a curve classified as a circle by Rata.SSR, we follow a similar procedure, except that angle of rotation is irrelevant. Once we have computed the bounding box, we take the average length of the two sides as the circle's diameter. Of course, it is not necessary to differentiate between circles and ellipses, but circles are more aesthetically pleasing; if a user intended to draw a circle it is preferable to preserve that intention where possible.

Figure 10 shows the formal Euler diagram obtained from the sketch in figure 3. The sketched curves c, d and e have been classified by Rata.SSR as circles and, therefore, rendered as circles in the formal diagram, whereas a and b have been identified as ellipses. Notice that b, in the sketch, is approximately parallel to the y-axis and has, therefore, been given a rotation of 90 degrees in the formal diagram. The computed angle of rotation of a in the sketch has, however, has been maintained in the formal diagram. We plan to extend the software so that it computes the abstract description of the formal diagram, which can be done using the algorithm described for the sketch. This will allow us to automatically compare the abstract description of the sketch and the formal diagram to highlight any differences that may have arisen during the formalization stage.

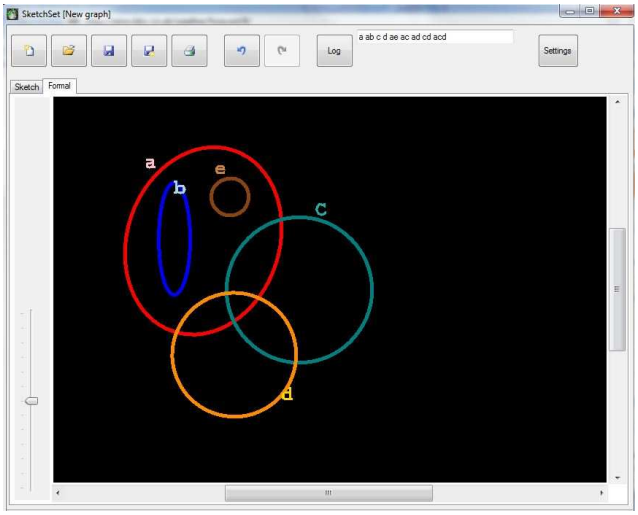

Figure 10. A formal Euler diagram generated from figure 3.

4. Conclusion

In this paper, we have presented the first techniques for recognizing sketches of Euler diagrams and developed a prototype implemtation. We have restricted to the case when all curves are circles or ellipses, which is often seen to be the case in user created Euler diagrams. However, there are collections of sets that cannot be accurately visualized using an Euler diagram drawn with ellipses and more arbitrary shaped curves must be used. We plan to extend this work to allow the recognition of arbitrary curves.

In order to produce quality diagrams from sketches drawn with arbitrary curves there are many challenges to be overcome. For instance, when a sketched curve has some symmetry, we should aim to preserve that symmetry in the formal diagram. In addition, we need to take into account approximate alignment and equality of spacing between sketched curves and preserve the perceived user intention in the formal diagram. The software should also incorporate more sophisticated editing functionality which the user can access in order to make modifications to both the sketch and formal diagram as required. We plan to write an Euler diagram sketch recognition plug-in for InkScape, so we can take advantage of the features already implemented in this freely available diagram editing tool.

The results presented here provide the necessary basis for developing sketch recognition software for notations that extend Euler diagrams with additional syntax. For example, constraint diagrams augment Euler diagrams with graphs, arrows, and shading. A simple example can be seen in figure 11; in this example, the curves of the underlying Euler diagrams do not intersect with each other but more complex diagrams can display arbitrary intersections between the curves. In constraint diagrams, the different style of node (asterisks versus filled circles) in the embedded graphs is of semantic importance and a sketch recognition engine for constraint diagrams would need to be sensitive to this difference.

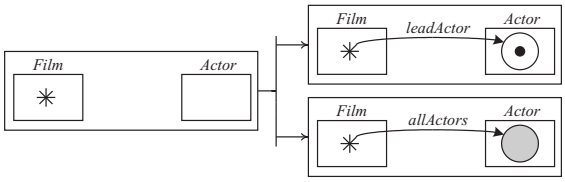

Figure 11. A constraint diagram.

Acknowledgement This research is supported by EPSRC grants EP/E010393/1, EP/H012311/1 and EP/H048480/1 and a Royal Society of New Zealand Marsden Grant.

References

[1] R. Blagojevic, P. Schmieder, and B. Plimmer. Towards a toolkit for the development and evaluation of sketch recognition techniques. In *Sketch Recognition Workshop*, 2009.

[2] S. Chang, B. Plimmer, and R. Blagojevic. Rata.ssr: Data mining for pertinent stroke recognizers. In *Sketch Based Interface Modeling*. ACM, 2010.

[3] R. DeChiara, U. Erra, and V. Scarano. VennFS: A Venn diagram file manager. In *Proceedings of Information Visualisation*, pages 120–126. IEEE Computer Society, 2003.

[4] L. Euler. Lettres à une Princesse d'Allemagne sur divers sujets de physique et de philosophie. *Letters*, 2:102–108, 1775.

[5] G. Farrell and W. Sousa. Repeat victimization and hot spots: The overlap and its implication for crime control and problem-oriented policing. *Crime Prevention Studies*, 12:221–240, 2001.

[6] V. Goel. *Sketches of thought*. MIT Press, 1995.

[7] G. Goldschmidt. *Visual and Spatial Reasoning in Design*, chapter The Backtalk of Self-Generated Sketches, pages 163–184. University of Sydney, 1999.

[8] J. Howse, F. Molina, J. Taylor, S. Kent, and J. Gil. Spider diagrams: A diagrammatic reasoning system. *Journal of Visual Languages and Computing*, 12(3):299–324, June 2001.

[9] S. Hughes. The Great British Venn diagram. http://qntm.org/uk, accessed July 30, 2010.

[10] E. Ip. Visualizing multiple regression. *Journal of Statistics Education*, 9(1), 2001.

[11] G. Johnson, M. Gross, and J. Hong. *Computational Support for Sketching in Design*. Now Publisher Inc., 2009.

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

[13] H. Kestler, A. Muller, H. Liu, D. Kane, B. Zeeberg, and J. Weinstein. Euler diagrams for visualizing annotated gene expression data. In *Proceedings of Euler Diagrams 2005*, Paris, September 2005.

[14] J. Landay and B. Myers. Interactive sketching for the early stages of user interface design. In *Chi 1995 Mosaic of Creativity*, pages 43–50, 1995.

[15] I. Oliver, J. Howse, G. Stapleton, E. Nuutila, and S. Törma. A proposed diagrammatic logic for ontology specification and visualization. In *International Semantic Web Conference*, 2009.

[16] B. Paulson and T. Hammond. Paleosketch: Accurate primitive sketch recognition and beautification. In *Intelligent User Interfaces*. ACM Press, 2008.

[17] B. Plimmer and I. Freeman. A toolkit approach to sketched diagram recognition. In *HCI*, pages 205–213. British Computer Society, 2007.

[18] B. Plimmer, H. Purchase, and H. Laycock. Preserving the hand-drawn appearance of graphs. In *Visual Languages and Computing*, pages 347–352, 2009.

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

[20] J. Soriano, K. D. B. Coleman, G. Visick, D. Mannino, and N. Pride. The proportional Venn diagram of obstructive lung disease. *Chest*, 124:474–481, 2003.

[21] N. Swoboda and G. Allwein. Heterogeneous reasoning with Euler/Venn diagrams containing named constants and FOL. In *Proceedings of Euler Diagrams 2004*, volume 134 of *ENTCS*. Elsevier Science, 2005.

[22] J. Thièvre, M. Viaud, and A. Verroust-Blondet. Using Euler diagrams in traditional library environments. In *Euler Diagrams 2004*, volume 134 of *ENTCS*, pages 189–202. ENTCS, 2005.

[23] J. Venn. On the diagrammatic and mechanical representation of propositions and reasonings. *Phil.Mag*, 1880.