Curve-based diagram specification and construction

Gennaro Costagliola*, Mattia De Rosa*, Andrew Fish†, Vittorio Fuccella* and Rafiq Saleh†
†School of Computing, Engineering&Mathematics, University of Brighton, UK. Email:{Andrew.Fish,R.Saleh}@brighton.ac.uk
*Department of Informatics, University of Salerno, Italy. Email:{gencos, matderosa, vfuccella}@unisa.it

***Abstract*—We present a code which captures the topology of closed-curve based diagrams (e.g. Euler diagrams (EDs) are used to visualize set-based relationships, whilst knot diagrams represent knotted structures), often used in visual languages. We briefly indicate how to construct a diagram from such a code. Furthermore, we present an interactive software artefact with a human-centric focus, permitting users the freedom to construct and transform diagrams at either the diagram or the code level, implementing algorithmic procedures developed.**

I. Introduction

EDs *with items* [1] visualise elements within the regions (set intersections) determined by the curves (representing sets), and have been for resource management, network visualisation and for display of search query results [1]–[3]. The additional constraint of (approximate) area-proportionality of regions to indicate relative sizes of sets has been used in bio-informatics for the representation of genetic set relations and for statistical data representation [4]–[6].

We propose that the import of significant machinery from knot theory to EDs is likely to be of great benefit to the development of the field, with the adoption of code-based approaches enabling use (a) as a specification language; (b) for diagram construction and interchange; (c) for diagram property identification; (d) for diagram transformations simulated with string rewriting techniques.

II. Diagram and Code Definitions

Definition 2.1: Let $\mathcal{C} = \{C_1, \ldots, C_n\}$ be a (ordered) family of immersed closed curves in the plane. Let X_i and Y_i denote the interior and exterior of C_i, respectively, for each $i \in I = \{1, \ldots, n\}$. Let each subset I' of I be called an *abstract zone* which *describes* the set $\bigcap_{i \in I'} X_i \cap \bigcap_{i \in I - I'} Y_i$, called a *concrete zone* of $\mathcal{C}$. We say $\mathcal{C}$ is *generic* if at most two curves intersect at any point, and all intersections are transverse, is *simple* if each C_i is simple (so has no points of self-intersection), has *connected zones* if the non-empty concrete zones of $\mathcal{C}$ are connected. A *link diagram* is a generic set (i.e. ignoring the ordering) of immersed closed curves, together with additional information of over/under at the crossings. A *wellformed ED* [7] or *simple ED* [8] is a simple generic family of immersed closed curves in the plane which has connected zones (noting the obvious correspondence between labelling of curves and an ordering); relaxing simplicity yields EDs.

We consider ordered families of generic immersed closed curves in the plane. Examples are provided in Figure 1, where the obvious correspondence between labelling of curves (e.g. using A, B, C) and curve ordering is assumed.

We consider Gauss paragraphs (previously used for links, with the term Gauss code or word used for single component knots), but with an additional ordering on the words. Fix an alphabet $\mathbb{N} \times \{+, -\}$, and abbreviate $(k, +)$ by k^+ and $(k, -)$ by k^-. Given a set of words $W = \{w_1, \ldots, w_n\}$, and a symbol j, let $|\, w_i \,|$ denote the length of the word w_i, and $|\, W \,|_j$ the number of occurrences of j in W. The ordering on the words in $W = \{w_1, \ldots, w_n\}$, is presented using the index.

Definition 2.2: An *ordered Gauss paragraph* (OGP) is an ordered set of words $W = \{w_1, \ldots, w_n\}$ over the alphabet $\mathbb{N} \times \{+, -\}$ such that:

- $|w_1| + |w_2| \ldots + |w_n|$ is even, and
- for every $i \in \mathbb{N}$ either $|W|_{i^+} = |W|_{i^-} = 0$ or $|W|_{i^+} = |W|_{i^-} = 1$.

Definition 2.3: An OGP of an Euler diagram $\mathcal{C} = \{C_1, \ldots, C_n\}$ with k-crossings is constructed as follows: assign a unique label l in $\{1, \ldots, k\}$ to each crossing and orient all curves; for each curve C_i, choose an arbitrary base point p_i on C_i which is not a crossing point; for each $i \in \{1, \ldots, n\}$, compute w_i by traversing the curve C_i once, starting and ending at the base point p_i, following the orientation of the curve, recording the label of each crossing met in turn, together with the sign associated with the strand that is met at that crossing. Within the traversal of C_i, the *sign* of crossing l is determined by the orientation of C_i relative to the orientation of the *passing strand* (i.e. the part of the curve, C_j, that it crosses) at l, as shown in Figure 2.

The diagram at the bottom of Figure 2 has two curves oriented clockwise. Each arc between a pair of crossing points is called a *segment*, with the corresponding pair of symbols in the code called a *segment code*. For instance, whilst traversing curve A, the segment which is the top arc of the diagram has segment code (3^+4^-); we use $(3^+4^-)^r$ to refer to the same segment, but in *reverse* (i.e. against the orientation of the curve). The adjacent table shows the *region boundary codes* for each of the regions of the diagram, which are a means of identification of the individual regions.

III. Diagram Construction

We can construct a unique diagram, on the sphere, from an OGP. Thus any specification of choice of the infinite face (to be used for the stereographic projection) determines the diagram in the plane. Based on the work of Carter [9], we can directly construct a diagram from an OGP via a solution of the planarity problem for the considered class of generic curves. The main idea is to associate with a OGP a Carter surface, which is a combinatorial 2-cell complex[1], where vertices correspond to crossing labels, edges correspond to

[1]A topological space with cell structure consisting of 0-cells corresponding to vertices, 1-cells corresponding to edges and 2-cells corresponding to faces.

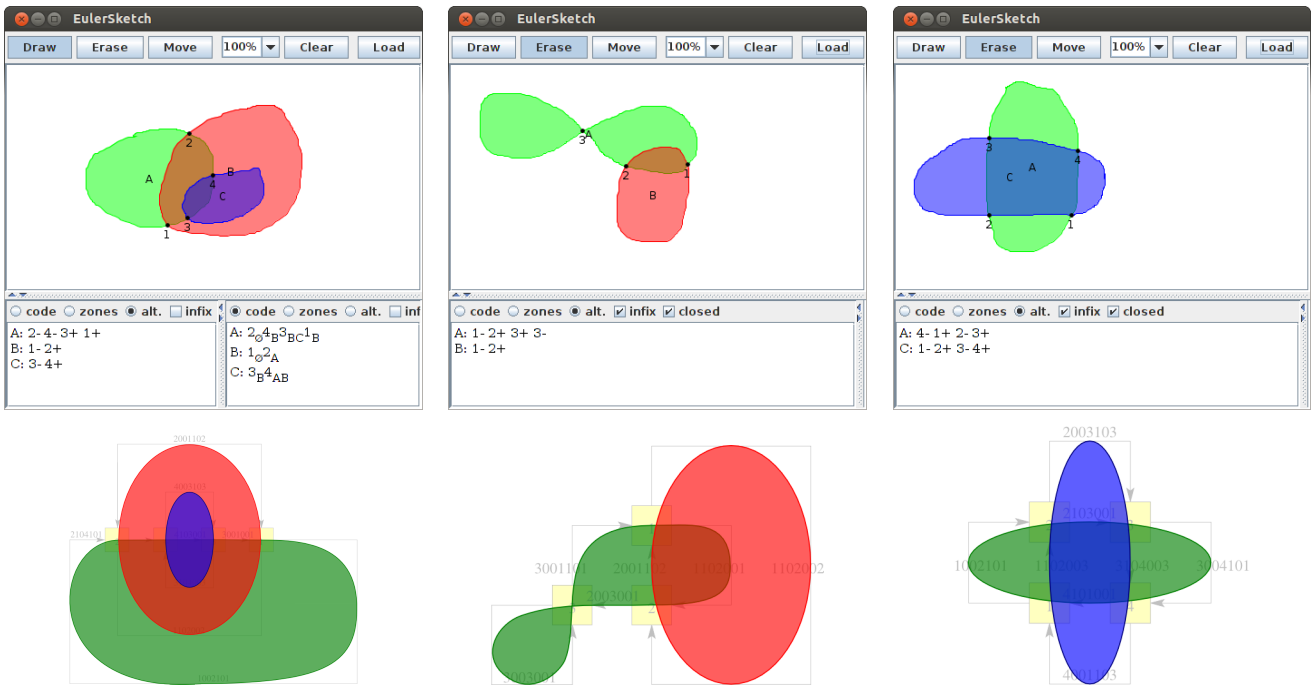


Fig. 1. From Left to Right: (1) is simple generic with connected zones; (2) is generic; (3) is simple generic; Top: sketched by a user of the tool, with code automatically computed; Bottom: diagrams generated from the code.

two consecutive labels and faces are found by successive left turns on each crossing (as illustrated in Figure 2) and then compute its Euler characteristics χ, which is the number of faces (cycles) minus the number of edges plus the number of vertices. If χ is equal to 2 then the OGP is planar. The OGP includes the information required to construct the Carter surface, and thus check planarity, whilst the embedding of the graph (the 1-skeleton of the 2-complex) in the plane such that the faces of the graph are the faces of the 2-complex (and such that the chosen face is unbounded) is precisely the required graph of the diagram.

Theorem 3.1: The OGP determines the topology of the diagram on the sphere.

We explicitly construct the combinatorial embedding of the graph of the diagram directly from the code, as indicated in Figure 2. The symbol for crossing i will appear exactly twice in the code, once with each choice of sign. So, suppose that the following strings of consecutive symbols appear in their containing words wi^+z and xi^-y, noting that the words are read cyclically and the signs of symbols w, x, y, z are omitted here. Then the clockwise order of the edges around the vertex for i is determined, as shown.

Theorem 3.2: The OGP, together with a specification of the infinite face, determines the topology of the diagram in the plane.

The choice of the infinite face can be specified in several ways, including as the region boundary code for the specified face. The options can be made available for user selection for initial generation, or simply preserved for diagram transformation, whilst preserving the mental map, for instance.

Theorem 3.3: Given an OGP of a diagram C, the set of region-boundary codes can be algorithmically computed, and from this, together with a choice of outer face, the set of abstract zones can be algorithmically computed.

We utilise an off-the-shelf algorithm for embedding the graph, producing a necessarily correct result. The outputs look extremely promising (e.g. see Figure 1), but further post-processing beautification could be used to enhance the output, and if topology preservation is a requirement (depends upon application domain) then one could make use of code preservation as a means of identifying topology preservation.

Since there are various methods to automatically produce EDs and to beautify them, and distinct methods will likely work well for certain tasks, it makes sense to ask if one can combine their capabilities, permitting users to choose which method they deem fit or to compare methods for their task. One can take any existing method or tool which produces diagram output (e.g. any existing generation or beautification method), and compute the code from the diagram (e.g. to pass an initial feasible layout from a generation method into another beautification method). Figure 1 shows diagram regeneration using the code taken from diagrams sketched by the user. Figure 3 highlights the similarities and differences of code changes in diagrams that differ by transformations, and shows the idea of the import of a diagram from another method, its encoding, followed by their regeneration from the code. A previous version of the prototype EulerSketch, using the static code instead of the gauss code, is described in [10].

IV. Related Work and Conclusion

In [11], alternative equivalent ED abstractions were developed, including the view as a *building sequence* of curve additions [12]. In [1], [13], efficient algorithms were provided for the online ED abstraction problem: given a concrete ED,

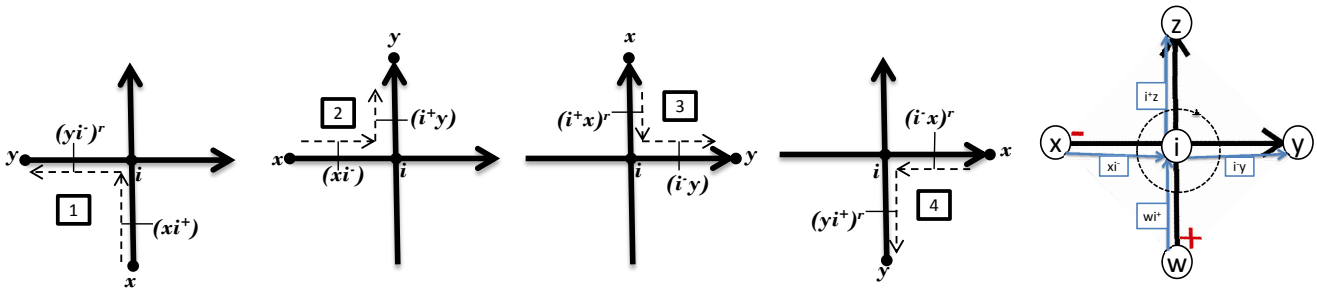
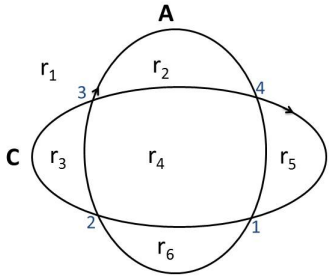


Fig. 2. Above – Left: The four traversal rules $t_1, \ldots, t_4$ indicate how to traverse the code corresponding to a traversal of a region of the diagram in which we turn left at every crossing. For example, t_1 shown on the left, is read as follows: if we approach the crossing i along the vertical strand the sign of i is $+$ and so we see xi^+ in the code (where x itself could be positive or negative). Turning left at the crossing corresponds to traversing segment yi^- (negative sign since that strand is negative) against its orientation, and so we obtain $(yi^-)^r$ in the code, where r indicates reversal orientation. Right: The direct computation of the combinatorial embedding via the clockwise ordering of edges around a vertex from an ordered Gauss paragraph. Below– a redrawing of (3) from Figure 1 used to indicate region-boundary codes, alongside the table of computed region boundary codes.



Region	Region boundary code
r_1	$\{(3^+4^-), (4^+1^-), (1^+2^-), (2^+3^-)\}$
r_2	$\{(3^-4^+), (3^+4^-)^r\}$
r_3	$\{(2^-3^+), (2^+3^-)^r\}$
r_4	$\{(1^-2^+)^r, (4^-1^+)^r, (3^-4^+)^r, (2^-3^+)^r\}$
r_5	$\{(4^+1^-)^r, (4^-1^+)\}$
r_6	$\{(1^+2^-)^r, (1^-2^+)\}$

compute the associated abstract ED and update this efficiently upon curve addition and removal. This utilised intersection points created in a building sequence, using these points (or equivalently curve segments) to mark zones. Checking membership of these marked points within regions can then be used to determine the new zone set quickly. By not utilising a graph based methodology within the interactive setting, the implementation becomes simple and computations efficient. In [14], the static code was introduced, which can be viewed as a human-centric abstraction of the computational concepts of [13]; the theory developed enabled the computation of the abstract zone set for simple generic families of immersed closed curves which were connected. Note that OGPs have a simpler syntax than the static code, requiring the inclusion of signs to indicate the relative orientation of curves at each crossing versus the explicit association of the set of all containing curves to each segment in the diagram. Previously, Clark [15] made use of java-area operations to check if the area is non-empty for every subset of the set of curves, whilst the polygon comparison algorithm of Weiler [16] enabled the computation of the 'concrete zones' (for their special case) in a more efficient manner. Previous work on ED sketch recognition [17] enabled the recognition of EDs, utilising a single stroke recogniser.

There are various existing diagram generation approaches, such as [8], [18]–[20], each of which will have its own advantages or disadvantages, providing aesthetically pleasing output for particular classes of diagrams, perhaps. In [21], they produce isocontours to highlight relationships amongst graph nodes within existing graph layout, useful in highlighting collections of venues within a map layout for instance. Applications in which graph nodes are fixed are prime example where the use of diagrams with multiple regions for single zones would be appropriate to avoid massive distortion of curves yielding unwieldy diagrams.

There have been many investigations of Gauss words [22], which were originally developed for knots, with questions of planarity (if there exists a realisation of a code as a generic immersed curve) paramount; based on the work of Carter in [9], Elton and Cairn presented an a combinatorial criteria in [23] which deals with the planarity problem of signed Gauss words. The specification of abstract EDs in terms of zone sets restricts the ability to consider the use of more general closed curves since it cannot capture variations in the topology of the diagram (in the sense that the language only permits the consideration of whether a zone is non-empty as opposed to comprised of multiple disconnected minimal regions, for example). We may utilise codes to explicitly capture the topology of an ED, giving a compact string-based means to specify an ED without requiring the provision of the entire diagram (as a graph say, or equivalently its plane dual graph). Since many visual languages with a foundation of sets of closed curves in the plane, this direction of research has clear potential for impact. Adopting a code-based approach is fundamental in that it opens up many new avenues of research, including: the investigation of the properties of codes (e.g. in relation to alterations of the wellformedness conditions), layout beautification via the incorporation of different planarity algorithms, or to improve the output from existing tools.

ACKNOWLEDGMENT

Partially supported by UK EPSRC, grant EP/J010898/ 1, Automatic Diagram Generation, and University of Salerno, grant "Cofinanziamento per attrezzature scientifiche e di supporto, grandi e medie (2005)".

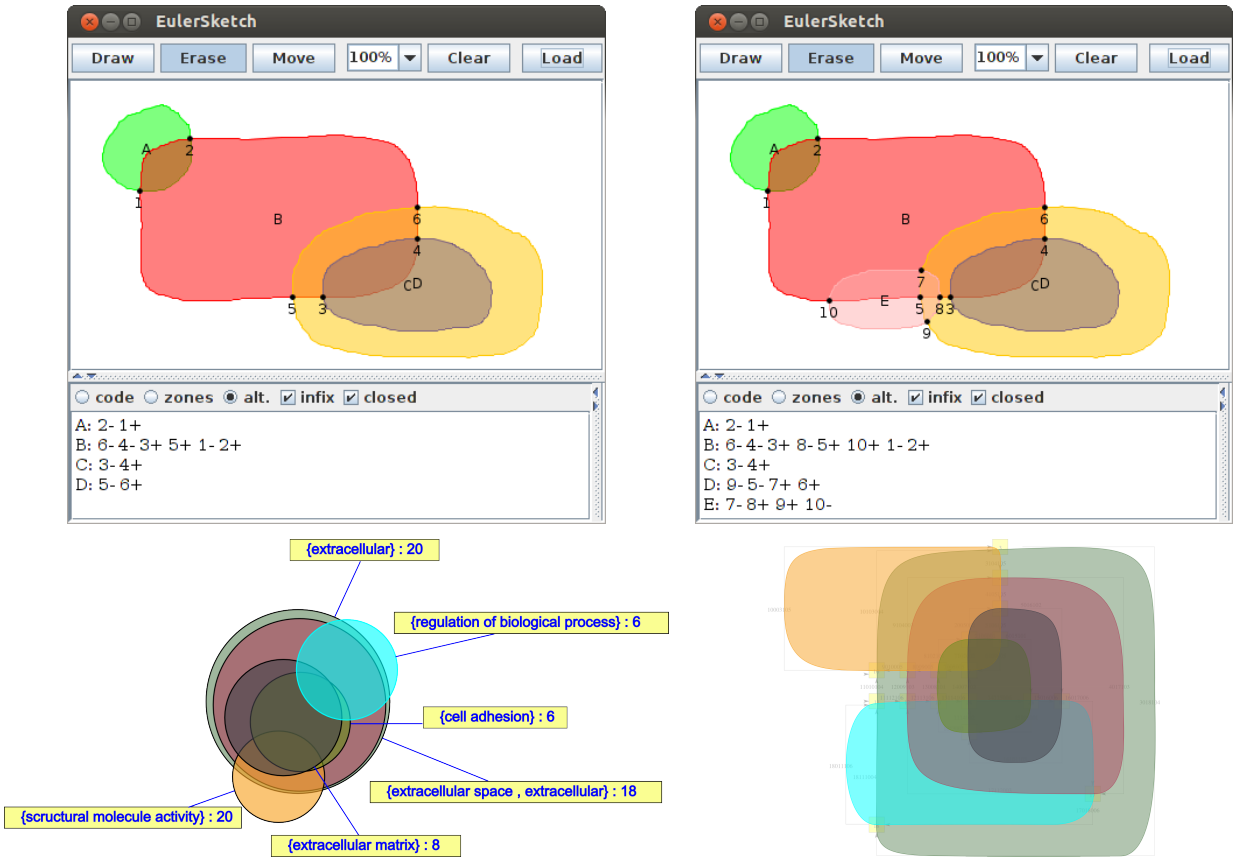


Fig. 3. Top – A pair of diagrams showing the effects of a user defined addition of a new curve; the codes are computed from the diagrams, but automation is possible via code based transformations; the OGP is shown beneath the diagrams. Bottom – Left: a diagram, similar to those in [4], where the zone sets are not clearly distinguishable. Right: an alternative view of the diagram based on our encoding and construction method. This view permits the identification of relationships more easily, whilst forgetting the area proportionality feature of the original; both views are useful for different tasks.

REFERENCES

[1] G. Cordasco, R. D. Chiara, and A. Fish, "Interactive visual classification with Euler diagrams," in *Proc. VL/HCC 2009*. IEEE, pp. 185–192.

[2] N. Riche and T. Dwyer, "Untangling Euler diagrams," *IEEE VCG*, vol. 16, no. 6, pp. 1090–1099, 2010.

[3] J. Thièvre, M. Viaud, and A. Verroust-Blondet, "Using Euler diagrams in traditional library environments," in *Proc. Euler 2004*, ser. ENTCS, vol. 134. ENTCS, 2005, pp. 189–202.

[4] H. Kestler, A. Müller, J. Kraus, M. Buchholz, T. Gress, H. Liu, D. Kane, B. Zeeberg, and J. Weinstein, "Vennmaster: Area-proportional Euler diagrams for functional GO analysis of microarrays," *BMC Bioinformatics*, vol. 9, p. 67, 2008.

[5] S. C. Chow, "Generating and drawing area-proportional Euler and Venn diagrams," Ph.D. dissertation, University of Victoria, 2007.

[6] L. Wilkinson, "Exact and approximate area-proportional circular venn and euler diagrams," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 2, pp. 321–331, 2012.

[7] J. Flower, A. Fish, and J. Howse, "Euler diagram generation," *Journal of Visual Languages & Computing*, vol. 19, no. 6, pp. 675–694, 2008.

[8] S. Chow, "Generating and drawing area-proportional euler and venn diagrams," Ph.D. dissertation, University of Victoria, 2007.

[9] J. Carter, "Classifying immersed curves," in *Proc. Amer. Math. Soc*, vol. 111, 1991, pp. 281–287.

[10] P. Bottoni, G. Costagliola, M. De Rosa, A. Fish, and V. Fuccella, "Euler diagram codes: interpretation and generation," in *Proc. VINCI 2013*. ACM, 2013. [Online]. Available: http://dx.doi.org/10.1145/2493102.2493116

[11] A. Fish and J. Flower, "Abstractions of Euler diagrams," in *Proc. Euler 2004*, ser. ENTCS, vol. 134, 2005, pp. 77–101.

[12] A. Fish, J. Flower, and J. Howse, "The semantics of augmented constraint diagrams," *JVLC*, vol. 16, pp. 541–573, 2005.

[13] G. Cordasco, R. D. Chiara, and A. Fish, "Fast region computations for reducible Euler diagrams," *Computation Geometry: Theory and Applications*, vol. 44, pp. 52–68, 2011.

[14] P. Bottoni, G. Costagliola, and A. Fish, "Euler diagram encodings," in *Proc. Diagrams '12*, pp. 148–162.

[15] R. Clark, "Fast zone discrimination," in *Proc. VLL 2007*, ser. CEUR, vol. 274, 2007, pp. 41–54.

[16] K. Weiler, "Polygon comparison using a graph representation." *Computer Graphics (SIGGRAPH '80)*, vol. 14, no. 3, pp. 10–18, 1980.

[17] M. Wang, B. Plimmer, P. Schmieder, G. Stapleton, P. Rodgers, and A. Delaney, "Sketchset: Creating euler diagrams using pen or mouse," in *Proc. VL/HCC 2010*. IEEE, 2010, pp. 75–82.

[18] P. Simonetto, D. Auber, and D. Archambault, "Fully automatic visualisation of overlapping sets," *Computer Graphics Forum*, vol. 28, pp. 967–974, 2009.

[19] G. Stapleton, P. Rodgers, J. Howse, and L. Zhang, "Inductively generating euler diagrams," *Transactions on Visualization and Computer Graphics*, vol. 17, no. 1, pp. 88–100, 2011.

[20] J. Flower, A. Fish, and J. Howse, "Euler diagram generation," *JVLC*, 2008.

[21] C. Collins, G. Penn, and S. Carpendale, "Bubble sets:revealing set relations with isocontours over existing visualisations," *IEEE Trans. Visual. and Computer Graphics*, vol. 15, no. 6, pp. 1009–1016, 2009.

[22] C. Gauss, "Werke," *Band 8. Teubner*, 1900.

[23] G. Cairns and D. Elton, "The planarity problem for signed Gauss words," *J. of Knot Theory and its Ramifications*, vol. 2, no. 4, pp. 359–367, 1993.