Computing All the Simple Symmetric Monotone Venn Diagrams on Seven Curves

by

Tao Cao
M Sc , University of Victoria, 1992

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

We accept this thesis as conforming
to the required standard

[redacted]

Dr F Ruskey, Supervisor (Department of Computer Science)

[redacted]

Dr J Ellis, Department member (Department of Computer Science)

[redacted]

Dr J Huang, Outside Member (Department of Mathematics and Statistics)

[redacted]

Dr J Phillips, External Examiner (Department of Mathematics and Statistics)

Supervisor Dr F Ruskey

Abstract

A family of intersecting simple closed curves (FISC) is a collection of simple closed curves in the plane with the properties that there is some open region common to the interiors of all the curves, and that every two curves intersect in finitely many points A normal FISC, or NFISC is a FISC with the properties that each curve touches the outer face and the family is convex-drawable In this thesis, we present a G-encoding technique for an NFISC, which generalizes a Grunbaum encoding for a simple monotone Venn diagram We prove that NFISC diagrams can be uniquely identified with their G-encodings, and simple monotone Venn diagrams can be identified with their Grunbaum encodings By applying this theory, we develop two algorithms to search all possible simple symmetric monotone (SSM) Venn diagrams with seven curves The total number of such diagrams is 23

Examiners

[redacted]

Dr F Ruskey, Supervisor (Department of Computer Science)

[redacted]

Dr J Ellis, Department member (Department of Computer Science)

[redacted]

Dr J Huang, Outside Member (Department of Mathematics and Statistics)

[redacted]

Dr J Phillips, External Examiner (Department of Mathematics and Statistics)

Contents

List of Figures

Acknowledgements

I wish to thank the following people who have contributed significantly to the thesis. First, my supervisor, Dr. Frank Ruskey, has shaped my understanding of the field. I thank him for his helpful advising and his kind assistance during the research in the past several years. Dr. John Ellis, from whom I have taken the course Algorithms, made some useful suggestions on improving the efficiency of the algorithms in Chpater 3 and Chapter 4. Dr. Jing Huang has done a number of corrections in the thesis. Dr. John phillips has taught me mathematics in Analysis, Algebras, Operator Theory, and Topology (and many more). I used some knowledge in Topology in the proof of the main theorem in this thesis. At last, but not least, I would like to express my gratitude to my wife, and my parents for their moral support and encouragement throughout my studies.

Chapter 1

Introduction

This thesis is concerned with simple symmetric monotone (SSM) Venn diagrams and their representations and related properties. It develops algorithms for generating and identifying all SSM Venn diagrams with seven curves.

The rest of this chapter gives fundamental concepts and introduces most of the terminology used later in the thesis. The chapter has four sections. The first section gives a concise introduction to the pertinent graph theory. In the second section we present some relevant topological graph theory, with which we will prove the main theorem of this thesis in Chapter 2. The third section talks about the basic definitions and theory of Venn diagrams, and the last section describes the contribution of the thesis.

In Chapter 2, we present several data structures for simple monotone Venn diagrams. G-encoding and Grünbaum encoding methods, matrix representations, and graph representations. These data structures are essential for generating, identifying and comparing Venn diagrams. Some detailed ex-

amples are studied. A number of properties and theorems regarding SSM Venn diagrams and these data structures are fully discussed and proved.

In Chapter 3 and Chapter 4, two algorithms for generating all the SSM Venn diagrams with seven curves are given. With the data structures for representing SSM Venn diagrams invented in Chapter 2, we show how all SSM Venn diagrams with seven curves are generated, how they are identified, and how they are compared with each other.

Conclusions and computational details are presented in Chapter 5. We find out that there are exactly 23 SSM Venn diagrams for $n = 7$.

1.1 The Basic Definitions of Graph Theory

In this thesis, all graphs are finite. An *undirected graph* $G = (V, E)$ consists of a finite nonempty set V and a set E of *unordered* pairs of distinct elements of V. The elements in V are called *vertices* of G, and the elements in E are called *edges* of G.

A *finite undirected multigraph* $G = (V, E)$ consists of a finite nonempty set V and a set E of unordered (*not necessarily distinct*) elements of V.

A *directed graph* $G = (V, E)$ consists of a finite nonempty set V and a set E of *ordered* pairs of (not necessarily distinct) elements of V. The elements in V are called *vertices* of G, and the elements in E are called *(directed) edges* of G.

A *plane graph* is a graph $G = (V, E)$ with the following properties:

1. $V \subseteq \mathbb{R}^2$,

2 every edge is an arc between two vertices,

3 every two edges are either disjoint or meet only at a common vertex

For a plane graph G, a *region* is an open set $U \subseteq \mathbb{R}^2 \backslash G$ such that for each pair u, v in U there exists a simple Jordan curve from u to v and that is contained in U. The *faces* of a plane graph G are the maximal regions of the plane that are disjoint from G. We also call faces regions if no confusion arises. We shall use $F(G)$ to denote the set of faces of G.

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. Then G_1 is *isomorphic* to G_2, written $G_1 \cong G_2$, if there is a bijective function $\phi : G_1 \longrightarrow G_2$ such that $uv \in E_1$ if and only if $\phi(u)\phi(v) \in E_2$.

A graph G is *planar* if it is isomorphic to some plane graph G'. The latter is called a *drawing* of G in the plane.

1.2 The Basic Definitions of Topological Graph Theory

1.2.1 Surfaces

In this thesis our interest is in representing a certain kind of graph on (some compact subset of) the plane, or more generally, on some *compact 2-dimensional manifold*. A set of points in a topological space is called a *2-manifold* if every point has a neighborhood which is homeomorphic to an open disk or *2-cell* $U^2 = \{x \in \mathbb{R}^2 : \|x\| < 1\}$[11]. A set of points in a topological space is *compact* if every covering with open sets has a finite subcovering.

A *surface* is a compact connected 2-dimensional manifold.

A surface is *non-orientable* if by following some closed loop on the surface, we can change clockwise rotations into anti-clockwise rotations. A surface is *orientable* if no such loops exist.

Spaces such as the plane and the Mobius strip are not surfaces by this definition. The plane is not compact, and the Mobius strip is not a 2-dimensional manifold. Spheres and tori are examples of orientable surfaces. Projective planes and Klein bottles are non-orientable surfaces.

1.2.2 Surfaces S_h and N_k

There are two ways we can obtain more complicated orientable or non-orientable surfaces: adding "handles" to a sphere, or adding "crosscaps" to a sphere.

To add a handle to the sphere, we first make two "holes" on a sphere, and orient their boundaries in the same directions. We then take a truncated cylinder, bend it into a shape of a handle, orient its ends in the same direction, and "glue" its boundaries to those on the sphere, preserving orientation. Then we have another orientable surface, which is equivalent to the torus. We can repeatedly add more handles to the sphere. A sphere with h handles is denoted by S_h, where h is called the *genus of the surface*.

To add a crosscap on a sphere, we make one "hole" on a sphere first, and "glue" the boundary by identifying each point with its opposite one. We can repeat this procedure k times to obtain a sphere with k crosscap. The

resulting surface is denoted by N_k

The classification theorem on surfaces states that [1]

Theorem 1.1 *An orientable surface is homeomorphic to S_h for some $h \geq 0$, a non-orientable surface is homeomorphic to N_k, for some $k > 0$*

1.2.3 Plane Models of Surfaces

Surfaces can be represented by polygons on the plane The polygons are called *plane models of the surfaces* [1] Since complicated surfaces are sometimes hard to draw in three-dimensions, plane models help us to represent surfaces in two-dimensions The edges of the polygons are assigned labels to indicate how the corresponding surfaces can be reassembled Two edges to be identified will be assigned the same labels and they are glued together along the same direction as indicated

For example, in Figure 1 1, the sphere is cut as indicated Its plane model is a quadrilateral, with labels a, a^{-1}, b, and b^{-1} In Figure 1 2, we show that the plane model of a double torus is

$$a_1,\ b_1,\ a_1^{-1},\ b_1^{-1},\ a_2,\ b_2,\ a_2^{-1},\ b_2^{-1}$$

In fact, as noted by White and Beineke [1], the orientable surface S_h can be formed from a $4h$-gon by identifying corresponding sides when the boundary labels are

$$a_1,\ b_1,\ a_1^{-1},\ b_1^{-1},\ \ldots,\ a_h,\ b_h,\ a_h^{-1},\ b_h^{-1}$$

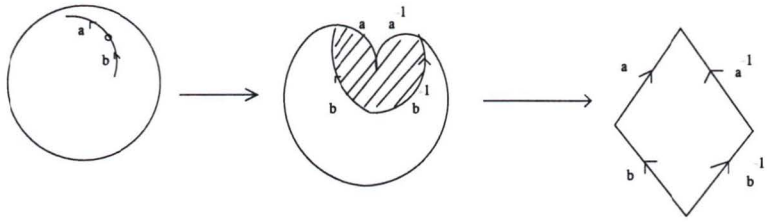


Figure 1 1 Polygon Representation of the Sphere

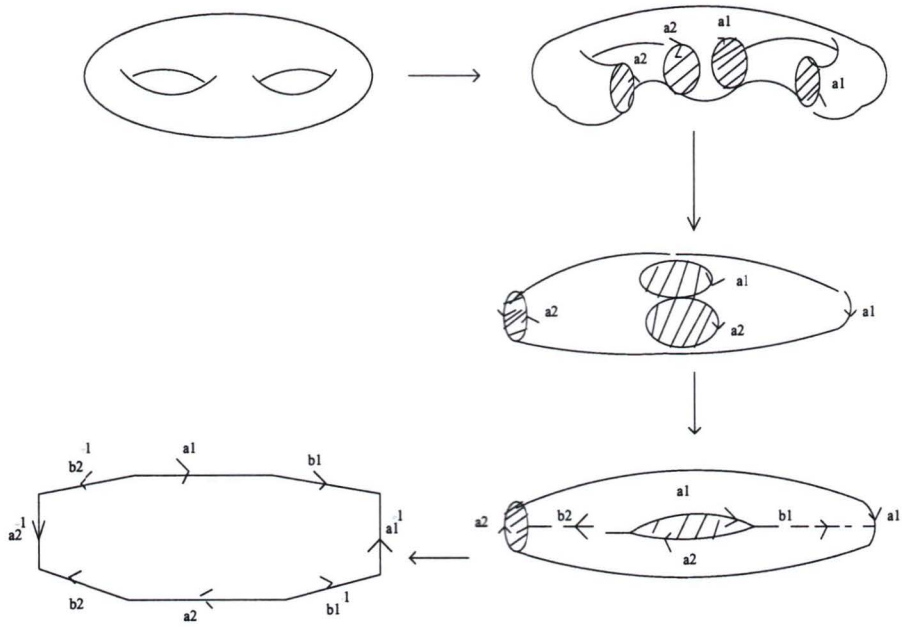


Figure 1 2 Polygon Representation of the Double Torus

Use of plane models also allows us to represent graphs on the plane, and could be a convenient tool to solve some surface-related problems. The examples presented here only illustrated the basic ideas, more details can be found in [1], [6] and [9].

1.2.4 Two-cell Embeddings

In the thesis, our particular interest lies in the embedding of Venn Diagrams in some surface (see Section 1.3). An *embedding* of a general graph $G = (V, E)$ in a surface S is a mapping τ of G to S such that

1. $\tau(V)$ are a set of points of S and $\tau(E)$ are a set of disjoint open arcs of S,
2. the distinct vertices of G are mapped to distinct points of S,
3. the edges of G are mapped to disjoint open arcs of S,
4. for any edge e joining vertices v and w of G, the open arc $\tau(e)$ joins the points $\tau(v)$ and $\tau(w)$ in S,
5. for any edge vw, and for any vertex u, $\tau(vw)$ does not contain $\tau(u)$ in S.

Moreover, if the complement of $\tau(G)$ relative to S is a collection of 2-cells, the embedding is called *2-cell embedding*. Indeed, the complement is necessarily a collection of open sets of S. The embedding is 2-cell only if all such open

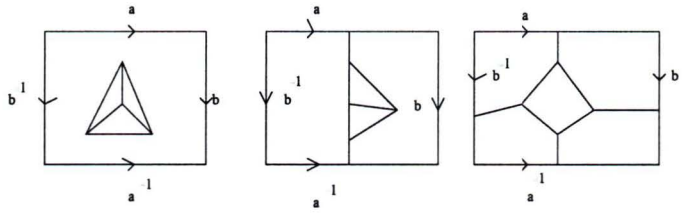


Figure 1.3 Embeddings of K_4 in a Torus

sets are homeomorphic to a disk. In Figure 1.3, three embeddings of K_4's in a torus, only (c) is a 2-cell embedding [1].

With S as a surface, the *Euler characteristic* $\chi(S)$ is defined by

$$\begin{aligned} \chi(S_h) &= 2-2h \quad \text{for the orientable surface } S_h, \\ \chi(N_k) &= 2-k \quad \text{for the non-orientable surface } N_k \end{aligned}$$

The following theorem is a generalization of Euler's polyhedron formula for the plane. It gives the relationship among the numbers of vertices, edges and regions in a 2-cell embedding.

Theorem 1.2 *[1] For a 2-cell embedding of a connected (general) graph with p vertices, q edges, and r regions in a surface S, we have*

$$p - q + r = \chi(S)$$

1.2.5 Rotation Scheme

Rotation scheme is a method "for describing a 2-cell embedding algebraically, so that no actual drawings need be made"[1]. Here we only represent this tool for 2-cell embeddings of finite graphs in orientable surfaces.

Let $G = (V, E)$ be a finite connected (multi)graph and assume $V = \{v_1, v_2, \ldots, v_n\}$. Each edge e has two oriented directions on it; we use $(v_i, v_j)_e$

to denote the oriented edge starting from v_i to v_j and $(v_j, v_i)_e$ for the one with opposite direction. For each vertex v_i, let E_i be the set of edges oriented from v_i, i.e.

$$E_i = \{(v_i, v_j)_e \mid e \in E \text{ for some } v_j \in V\}$$

Let Φ_i be the set of the cyclic permutations of E_i. Then there is a one to one correspondence between the set of 2-cell embeddings of G and the Cartesian product $\prod \Phi_i$.

Theorem 1.3 *[1] Let $G = (V, E)$ be a finite connected (multi)graph. Define E_i and Φ_i as above. Then each choice of permutations $(\phi_1, \phi_2, \cdots, \phi_n)$ of $\Phi_1 \times \Phi_2 \times \cdots \times \Phi_n$ determines a 2-cell embedding of G in some orientable surface S_h. Conversely, for any 2-cell embedding of G in S_h, there is a corresponding set of permutations which yields that embedding.*

Proof: For each permutation ϕ of the set E_i of the oriented edges at vertices v_i, $1 \leq i \leq n$, we can define a mapping $\hat{\phi}$ on set E_i, so that for each (v_i, v_j) for some j, $\hat{\phi}(v_i, v_j)_e$ is the oriented edge next to $(v_i, v_j)_e$ in the permutation ϕ of E_i.

Let E^* be the set of all the oriented edges of G. For the given choice of permutations $(\phi_1, \phi_2, \cdots, \phi_n)$ of $\Phi_1 \times \Phi_2 \times \cdots \times \Phi_n$, define a mapping Φ on E^* by

$$\begin{array}{lll} \Phi^* & E^* & \longrightarrow E^* \\ & (v_i, v_j)_e & \longrightarrow \hat{\phi}_j(v_j, v_i)_e \end{array}$$

This map is one to one and onto if $\hat{\phi}_{j_1}(v_{j_1}, v_{i_1})_{e_1} = \hat{\phi}_{j_2}(v_{j_2}, v_{i_2})_{e_2}$, the two edges should be in the same set of oriented edges at the vertex, i e $v_{j_1} = v_{j_2}$ By the definition of $\hat{\phi}_{j_1}$, $(v_{j_1}, v_{i_1})_{e_1} = (v_{j_2}, v_{i_2})_{e_2}$, is the same oriented edge after a common oriented edge adjacent to a common vertex $v_{j_1} = v_{j_2}$ So $v_{i_1} = v_{i_2}$ and $e_1 = e_2$ Therefore, Φ^* is a permutation of E^*

Each orbit of Φ, which is a closed walk of the graph, determines a region We then paste these regions together by identifying each oriented edge $(v, w)_e$ with $(w, v)_e$ Since each permutation ϕ is cyclic, the result of the pasting is a 2-manifold Since all edges are pasted, there are no boundaries for the 2-manifold, the 2-manifold indeed is a surface This surface can be oriented consistently such that at each vertex v_i, the oriented edge $(v_i, v_j)_e$ is followed by $\hat{\phi}_i(v_i, v_j)_e$ The surface is orientable hence it is a S_h for some $h \geq 0$ The number of regions, denoted by f, is the number of the orbits of Φ So the genus h can be calculated by Euler's formula

For each 2-cell embedding of G on S_h, it is trivial to obtain a choice of permutations $(\phi_1, \phi_2, \ldots, \phi_n)$ of $\Phi_1 \times \Phi_2 \times \cdots \times \Phi_n$ ∎

1.3 Venn Diagrams

A *Venn diagram* is a plane diagram consisting of n simple Jordan curves such that each of all possible intersections of the interior or the exterior of the curves is non-empty and connected There are at most 2^n possible intersections of the interior or exterior of the n curves

Before giving the formal definition of Venn diagrams, we should examine

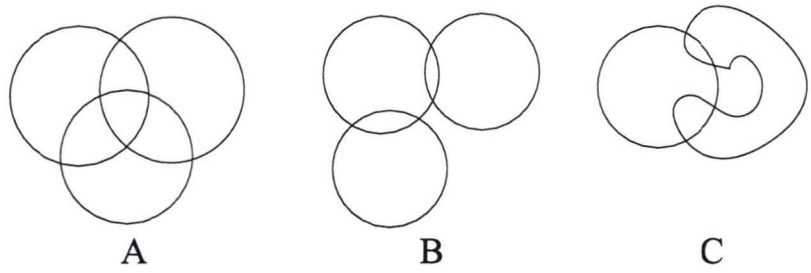


Figure 1 4 (A) 3-Venn Diagram, (B) not a Venn diagram,(C) not a Venn diagram

several examples

In Figure 1 4, the diagram (a) has all eight possible intersections of the interior or exterior of the three curves The eight possible intersections are

- one intersection which is inside all three curves,
- one intersection outside all three curves,
- three intersections which are inside two curves and outside the other curve,
- three intersections which are inside one curve and outside the other two

Furthermore, each intersection is in exactly one region The resulting diagram is a Venn diagram Diagram (b) is not a Venn diagram since the interior of all three curves do not exist Diagram (c) is not a Venn diagram either since the intersection of interiors of the two curves are separated into two regions

In 1880, British mathematician John Venn popularized Venn diagrams [13]. Here we follow Grünbaum [7] in the terminology.

Let $C = \{ C_1, C_2, \ldots, C_n \}$ be a collection of simple closed curves drawn in the plane. The collection C is said to be an *independent family* if the intersection of $X_1, X_2, \ldots, X_n$ is nonempty, where each X_i is either $\mathrm{int}(C_i)$ (the interior of C_i) or is $\mathrm{ext}(C_i)$ (the exterior of C_i).

Furthermore, if each such intersection is connected, then C is an *n-Venn diagram* (or just *Venn diagram*). "The condition that there are only a finite number of intersection points is usually assumed in the literature, but often not stated explicitly. It rules out segments of curves from intersecting." [12]

A Venn diagram is *simple* if no three curves intersect at a common point.

A Venn diagram is *monotone* if every k-region (the area inside k curves, $0 < k < n$) is adjacent to both a $(k-1)$-region and $(k+1)$- region.

A Venn diagram is *symmetric* if there is a point x about which the diagrams may be rotated by $2i\pi/n$ and remain invariant, for $i = 0, 1, \ldots, n-1$. A Venn diagram is *polar symmetric* if it is invariant under stereographic projection when one puts its center at the "north pole" of the sphere.

Two Venn diagrams are called *isomorphic* if they are topologically isomorphic. In this case, one of them can be changed into the other or its mirror image or its polar image by a homeomorphic transformation of the plane.

It is well known that if an n-Venn diagram is symmetric then n is prime [12], [5]. Symmetric Venn diagrams for $n = 2, 3, 5, 7$ have been found. For n is 2 or 3, the symmetric Venn diagrams are trivial. For $n = 5$, there

exists only one simple symmetric monotone Venn diagram [12]. For $n = 7$, there are many simple or non-simple symmetric Venn diagrams have been found. For $n = 11$, Hamburger [10] discovered a method for drawing a non-simple symmetric 11-Venn diagram. However, no simple symmetric 11-Venn diagrams have been found yet. The total number of such simple or non-simple symmetric n-Venn diagrams for an arbitrary prime number n is still unknown. The purpose of this thesis is to determine the total number of simple symmetric monotone 7-Venn diagrams.

Since the 1980's, people have found that symmetric Venn diagrams have close relationships with some other interesting topics in computer science, such as single-track Gray codes. Symmetric Venn diagrams also provide good challenges for people drawing graphs. However, not much is known about these kind of diagrams yet. The sufficient and necessary conditions about the existence of the symmetric Venn diagrams for a prime number n are still unknown. Now some mathematics and computer science experts are using computers to generate Venn diagrams to try and find out all symmetric Venn diagrams for some small prime numbers. Investigation of these diagrams is ongoing in order to find out more properties and interesting results. New achievements may be helpful to explore the sufficient and necessary conditions mentioned above.

Since 1996 23 simple symmetric monotone (SSM) Venn diagrams for n=7 have been discovered, where 6 of them are polar symmetric. A. Edwards found the first 5 SSM Venn diagrams with polar symmetry, one of which

was also found by B Grünbaum F Ruskey found another 18 SSM Venn diagrams, one of them with polar symmetry Now the question is are there any more of them? To answer this question, one can study the properties of SSM Venn diagrams for n=7, and write a computer program to do an exhaustive search

The program should find all SSM Venn diagrams which have been found before by other people It should also do exhaustive checking on all possible cases to see whether there are additional SSM Venn diagrams If there are such diagrams, the program should print out the diagrams' structure

1.4 Contribution of the Thesis

This thesis discusses several data structures such as 0-1 matrices, Grünbaum encoding, and graph representation for SSM Venn diagrams A number of properties and theorems regarding these data structures and SSM Venn diagrams are discovered and proved This thesis also presents two algorithms for computing all SSM 7-Venn diagrams, one using Grünbaum encoding, and the other using graph representation to identify the diagrams Grünbaum encoding is an encoding method for identifying a SSM Venn diagram in practice However, the conjectured one-to-one correspondence between the set of Grünbaum encoding and the set of SSM Venn diagrams had never been proved

The major achievement of this thesis is that it proves Grünbaum encodings do indeed identify SSM Venn diagrams In this thesis, a new encoding

method, G-encoding, for diagrams consisting of a certain type of Jordan curves (called *normal diagrams*), as a general case of Grünbaum encoding and SSM Venn diagrams, is studied. The one-to-one corresponding relationship between G-encoding and such normal diagrams is proved. As a result we prove that the Grünbaum encoding, which is a special case for SSM Venn diagrams , is a representation method for simple monotone Venn diagrams.

For this study, two computer programs were implemented to do exhaustive searching for all SSM 7-Venn diagrams, using Grünbaum encoding or graph representation respectively to identify the diagrams. We verified that the total number of diagrams is 23, and the two identifying methods give us exactly the same solutions.

Chapter 2

Representations of Symmetric Monotone Venn Diagrams

2.1 G-encoding Representation

2.1.1 The Definition and Properties of G-encoding

The G-encoding technique is for graphs consisting of a normal collection of Jordan curves. For simple symmetric monotone (SSM) Venn diagrams, G-encoding has its special case Grünbaum encoding.

Definition 2.1 *[2]* A family of intersecting simple closed curves (FISC) is a collection of simple closed curves in the plane with the properties that there is some open region common to the interiors of all the curves, and that every two curves intersect in finitely many points.

Definition 2.2 A *normal* FISC, or NFISC, is a FISC satisfying the following conditions:

1. every curve contributes to the boundary of the infinite face (the outer

most face),

2 the collection is simple, i e , at every point of intersection exactly two curves meet, and they cross each other,

3 the collection is convex drawable, i e , there is a FISC $\mathcal{C}$ whose curves are all convex and the collection can be transformed into $\mathcal{C}$ by a homeomorphic transformation of the plane

Let $\mathcal{F}$ be a FISC, and let $G(\mathcal{F})$ be a plane graph, where its vertices are the intersections of $\mathcal{F}$, and its edges are the curve arcs between vertices The directed dual $\vec{D}(\mathcal{F})$ of $G(\mathcal{F})$ is the dual graph of $G(\mathcal{F})$ with edges oriented to indicate inclusion in fewer interiors of the curves In [2], it is shown that $\mathcal{F}$ is convex drawable if and only if $\vec{D}(\mathcal{F})$ contains only one source and only one sink

Let $\mathcal{C}$ be an NFISC of n Jordan curves, and call the diagram consisting of these n curves an *n-diagram* The n Jordan curves are labeled in the following way Arbitrarily choose a curve as curve 0, and Start at a segment of curve 0 where it touches the outer most region, (which will be labeled as segment 0 of the curve 0), we label the next $n-1$ curves by their clockwise appearance on the out most region

Let M_n (or just M) be the number of curve segments touching the outer face Furthermore, for each curve, we divide it into $2k$ oriented segments if the curve touches the outer face k times the curve is partitioned by the endpoints of the segments which touch outer face, and the segment 0 of the

curve is the one which touches the outer face and is the first segment of the curve we encountered when we label the curve. Each segment is oriented clockwise. For our convenience, we define that each endpoint belongs to the segment with it as the starting point. A *G-encoding* of the n-diagram consists of $M+1$ sequences and an $n \times n$ matrix P. The first sequence, call it I, is an outer face intersection list of length M. Starting at curve 0, it goes clockwise around the outer face recording the curves encountered.

$$I = \{c_0, c_1, \quad , c_{M-1}\},$$

where $c_i \in \{0, 1, \quad , n-1\}$. The other M sequences $\{w_i\}_{i=0}^{M-1}$ record intersections along a given curve with other curves, starting at the segment where it touches the outer face. For each intersection, we record the encountered curve's number and the segment number of the encountered curve in the form of (curve number, segment number).

$$\begin{aligned} w_0 &= w_{00}\ w_{01} \quad w_{0m_0} \\ w_1 &= w_{10}\ w_{11} \quad w_{1m_1} \\ &\vdots \\ w_{M-1} &= w_{M-1,0}\ w_{M-1,1} \quad w_{M-1,m_{M-1}}, \end{aligned}$$

where each w_{ij} is a pair of a curve number and a segment number. Without loss of generality, we assume that c_i and c_j in I are for curve i and curve j respectively, and assume curve i intersects with curve j exactly k times, then we have k pairs of $(j, \star)$ in w_i string.

$$w_i = \quad (j, \star) \quad (j, \star) \quad \cdots$$

We use $P[i][j]$ to record the order of the pair $(j, \star)$ which is corresponding to the first intersection on curve j, among all the intersections of curve i and

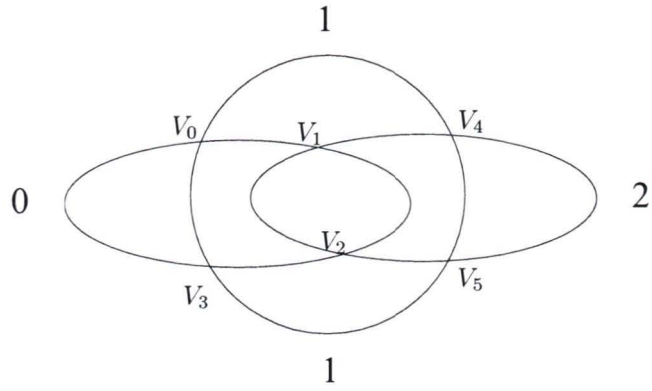


Figure 2.1: A Simple Example of an NFISC

curve j, along curve j starting at segment 1 of curve j. We define $P[i][j] = m$ if the pair of $(j, \star)$ is corresponding to w_{im} in w_i.

Example 2.3 Figure 2.1 shows an NFISC diagram consists of three curves. We label the curves with label 0, 1 and 2 as indicated in the figure. We also label all vertices as v_0, v_1, ..., v_5, to illustrate its G-encoding. Curve 0, curve 1 and curve 2 are divided into 2, 4 and 2 segments respectively.

curve 0
- segment 0: arc $[v_3, v_0)$
- segment 1: arc $[v_0, v_1, v_2, v_3)$

curve 1
- segment 0: arc $[v_0, v_4)$
- segment 1: arc $[v_4, v_5)$
- segment 2: arc $[v_5, v_3)$
- segment 3: arc $[v_3, v_0)$

curve 2
- segment 0: arc $[v_4, v_5)$
- segment 1: arc $[v_5, v_2, v_1, v_4)$

The G-encoding of Figure 2.1 is

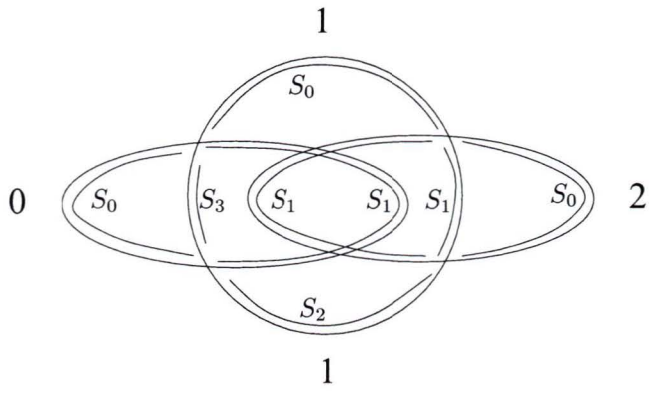


Figure 2 2 Segment Labels

$$\begin{array}{ll} 0 & w_0 = (1,\,0)\ (2,\,1)\ (2,\,1)\ (1,\,3) \\ 1 & w_1 = (2,\,0)\ (2,\,1)\ (0,\,0)\ (0,\,1) \\ 2 & w_2 = (1,\,2)\ (0,\,1)\ (0,\,1)\ (1,\,1) \\ 1 & w_3 = (0,\,0)\ (0,\,1)\ (2,\,0)\ (2,\,1) \end{array}$$

where I is

$$\{c_0, c_1, c_2, c_3\} = \{0, 1, 2, 1\}$$

Since curve 0 and curve 1 intersects at v_0 and v_3, and v_3 is prior to v_0 along curve 1 starting at segment 1, we have $P[0][1]$ equals to the position of v_3 in w_0, which is the (second) index 3 of $w_{03} = (1, 3)$ in string w_0 Similarly we can figure out the value for all other $P[i][j]$'s

$$\begin{array}{l} P[0][1] = 3,\ P[0][2] = 2 \\ P[1][0] = 3,\ P[1][2] = 1 \\ P[2][0] = 2,\ P[2][1] = 3 \end{array}$$

We may simply write the G-encoding into one table as

0	1	2	[2]	[1]
1	2	[2]	0	[0]
2	1	0	[0]	[1]
1	0	[0]	2	[2]

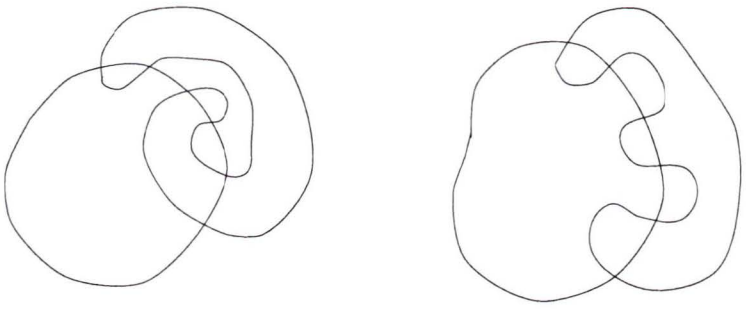


Figure 2 3 A Counter-example for Convex Drawable

Example 2 4 The G-encoding of Victoria (Figure 2 9) is

0 1 4 2 5 3 6 [1] 6 3 5 3 6 [2] 5 1 6 1 5 [3] 6 2 5 1 [4] 2 6 1 6 2 [5] 1 4 2 4 1 [6]
1 2 5 3 6 4 0 [2] 0 4 6 4 0 [3] 6 2 0 2 6 [4] 0 3 6 2 [5] 3 0 2 0 3 [6] 2 5 3 5 2 [0]
2 3 6 4 0 5 1 [3] 1 5 0 5 1 [4] 0 3 1 3 0 [5] 1 4 0 3 [6] 4 1 3 1 4 [0] 3 6 4 6 3 [1]
3 4 0 5 1 6 2 [4] 2 6 1 6 2 [5] 1 4 2 4 1 [6] 2 5 1 4 [0] 5 2 4 2 5 [1] 4 0 5 0 4 [2]
4 5 1 6 2 0 3 [5] 3 0 2 0 3 [6] 2 5 3 5 2 [0] 3 6 2 5 [1] 6 3 5 3 6 [2] 5 1 6 1 5 [3]
5 6 2 0 3 1 4 [6] 4 1 3 1 4 [0] 3 6 4 6 3 [1] 4 0 3 6 [2] 0 4 6 4 0 [3] 6 2 0 2 6 [4]
6 0 3 1 4 2 5 [0] 5 2 4 2 5 [1] 4 0 5 0 4 [2] 5 1 4 0 [3] 1 5 0 5 1 [4] 0 3 1 3 0 [5]

Example 2 5 If the collection of curves are not normal, then G-encoding cannot determine a diagram uniquely. In Figure 2 3, two diagrams, which are not normal, have the same G-encoding

0 1 1 1 1 1 [1]
1 0 0 0 0 0 [0]

However, they are not isomorphic

and in Diagram B, curve 1 intersects curve 0 in the following order

5, 4, 3, 2, 1, 0

None of the last two sequences can be obtained by circular shifting the first one

Theorem 2.9 *(Ruskey) Each G-encoding of an NFISC of n Jordan curves uniquely determines a 2-cell embedding of the n-diagram in some sphere* S_0

Proof Given an n-diagram of NFISC with n curves, it is trivial to construct a G-encoding. The difficult part of the proof is showing that, given a G-encoding of an n-diagram of NFISC, the diagram can be determined and it is unique up to homeomorphic transformation of the plane. Our strategy is to show that the G-encoding determines a 2-cell embedding of the diagram (as a planar graph) onto the sphere, and thus that uniqueness is guaranteed by Theorem 1.3. The desired n-diagram can be obtained via a suitable stereographic projection from the sphere to the plane.

Assume the given G-encoding is I, $\{w_i\}_{i=0}^{M-1}$, $\{P[i][j]\}_{i,j=0,i\neq j}^{M-1}$, and assume string $I = \{c_0, c_1, \ldots, c_{M-1}\}$, which is a (circular) list consisting of 0, 1, $\ldots$, $n-1$ with repetition allowed. Each string w_i actually records the curve numbers with which curve i intersects consequently, starting at a point where it touches the outer face. Furthermore, $P[i][j]$ tells us that which intersection on curve i is the one that curve j intersects with curve i after curve j touches the outer face first time.

OBTAIN_NORMAL_GRAPH_REPRESENTATION

Input vertex u touches most outside region
vertex v touches most inside region

1 $k \leftarrow 0$
2 for each neighbor w of u touches the most outside region
3 TRAVERSAL_GRAPH(w, u, leftmost)
4 Update($G[k]$), $k \leftarrow k+1$
5 TRAVERSAL_GRAPH(w, u, rightmost)
6 Update($G[k]$), $k \leftarrow k+1$
7 for each neighbor w of v touches the most inside region
8 TRAVERSAL_GRAPH(w, v, leftmost)
9 Update($G[k]$), $k \leftarrow k+1$
10 TRAVERSAL_GRAPH(w, v, rightmost)
11 Update($G[k]$)
12 lexicographic sort $G[0]$, $G[1]$, . . . , $G[7]$

Figure 4 4 Algorithm for Normal Graph Representations

TRAVERSAL_GRAPH

Input integer label of vertex f from vertex f
integer label of vertex t to vertex t
direction leftmost or rightmost

```
1   from ← f, to ← t
2   n ← 0
3   for i = 0 to 125
4       to ← ( to+ i ) mod 126
5       if vertex labled to is WHITE
6           DFS(from, to, direction)
7           from ← to
```

Figure 4.5 Algorithm for Traversal of a Graph

```
DFS

Input: label of vertex from
       label of vertex u
       direction: leftmost or rightmost

1.  if color of vertex u is BLACK, return
2.  color of u ← BLACK
3.  F[u] = n
4.  n ← n + 1
5.  if direction = leftmost version
6.     for all unvisited vertex v adjacent to u in clockwise order
7.        DFS( u, v, leftmost)
8.  else if direction = rightmost
9.     for all unvisited vertex v adjacent to u in counter-clockwise order
10.       DFS( u, v, rightmost)
```

Figure 4.6 Algorithm of Leftmost and Rightmost DFS

We can compare the first normal graph representation of V with that of the SSM 7-Venn diagrams we have ready obtained, to see whether V is a new diagram.

Chapter 5

Conclusion

We have implemented two computer programs in order to compute all the simple symmetric monotone (SSM) Venn diagrams with seven curves. One program uses Grünbaum encoding to identify the diagrams (Algrithm One, Chapter 4) and the other program uses graph representations to identify the SSM diagrams (Algorithm Two, Chapter 4). In each program, there are 20,125,224 matrices generated. The total number of different SSM Venn diagrams we obtained is 23. The two programs come out with the exactly the same solutions. These 23 SSM diagrams match with the 23 SSM Venn diagrams found by A. Edwards, B. Grünbaum and F. Ruskey. These findings verified in practice that the Grunbaum encoding is indeed a representation of an SSM Venn diagram, which we have proved in theory in Chapter 2.

Index of Solutions	Index of Matrices
1	96,948
2	144,637
3	145,004
4	150,449
5	245,226
6	270,995
7	337,325
8	420,957
9	487,352
10	650,806
11	740,224
12	746,788
13	746,888
14	751,414
15	1,097,622
16	1,126,308
17	1,349,320
18	1,467,557
19	2,218,590
20	2,219,552
21	2,247,989
22	2,314,011
23	2,782,864

Bibliography

[1] Arthur T. White, Lowell W. Beineke. "Topological Graph Theory," *Selected Topics in Graph Theory*, ed. Lowell W. Beineke, Robin J. Wilson, Academic Press, 1978.

[2] B. Bultena, B. Grunbaum, F. Ruskey. "Convex Drawings of Intersecting Families of Simple Closed Curves," preprint.

[3] T. Cormen, C. Leiserson, R. Rivest. *Introduction to Algorithms*, McGraw-Hill Book Company, 1990.

[4] R. Diestel. *Graph Theory*, Graduate Texts in Mathematics 173, Springer-Verlag 1995.

[5] A. W. F. Edwards. "Seven-set Venn Diagrams with Rotational and Polar Symmetry," *Combinatorics, Probability, and Computing*, 7 (1998) 149-152.

[6] P. A. Firby, C. F. Gardiner. *Surface Topology*, second edition, Ellis Horwood Limited, 1991.

[7] B Grünbaum "Venn Diagrams and Independent Families of Sets," *Mathematics Magazine*, Jan-Feb 1975, 13-23

[8] B Grünbaum "On Venn Diagrams and the Counting of Regions," *The College Mathematics Journal*, 15 (1984) 433-435

[9] H B Griffiths *Surfaces*, second edition, Cambridge University Press, 1981

[10] P Hamburger "Doodles and Doilies," preprint

[11] W S Massey *Algebraic Topology An Introduction*, Harcourt Brace Jovanovich, New York, 1967

[12] F Ruskey "A Survey of Venn Diagrams," *Electronic Journal of Combinatorics*, 4 (1997) DS#5

[13] J Venn "On the diagrammatic and mechanical representation of propositions and reasonings," *The London, Edinburgh, Dublin Philosophical Magazine and Journal of Science*, 9 (1880) 1 - 18

VITA

Surname Cao Given Names Tao

Place of Birth Wuhan, P R China

Educational Institutions Attended

University of Victioria	1990 to 2000
Wuhan University	1982 to 1986

Degree Awarded

M Sc	University of Victoria	1992
B Sc	Wuhan University	1986

Honours and Awards

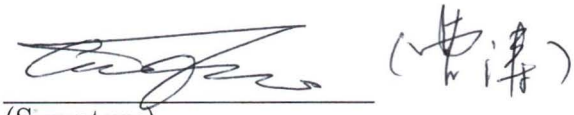
University of Victoria Fellowship	1990-1995
People's Prize	1982-1986

PARTIAL COPYRIGHT LICENSE

I hereby grant the right to lend my thesis to users of the University of Victoria Library, and to make single copies only for such users or in response to a request from the Library of any other university, or similar institution, on its behalf or for one of its users. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by me or a member of the University designated by me. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Title of Thesis:

Computing All the Simple Symmetric Monotone Venn Diagrams on Seven Curves



Author: (Signature)

TAO CAO
(Name in Block Letter)

Jan 2, 2002
(Date)

Assume in string I, the first occurrence of digit 0, 1, ..., $n-1$ is at i_0, i_1, ..., i_{n-1}, (i e , let x be the smallest index for a given k $(0 \leq k \leq n-1)$ such that $c_x = k$, then denote x as i_k) Now we label all vertices (intersections of curves) consecutively along string w_{i_k} (i e , along curve k) for $0 \leq k \leq n-1$ We will finish the labeling process in the following two steps

Step one Starting with w_{i_0}, where

$$w_{i_0} \quad w_{i_0 0}\ w_{i_0 1} \quad w_{i_0 m_0},$$

we label the vertex corresponding to $w_{i_0 0}$ as vertex $v_0 = 0$, $w_{i_0 1}$ as $v_1 = 1$, and so on We then have the following result for $w_{i_0 0}$

$$w_{i_0} \quad w_{i_0 0}(v_0)\ w_{i_0 1}(v_1) \quad w_{i_0 m_0}(v_{m_0})$$

For w_{i_1} we label all intersections of curve 1 with other curves except curve 0 In general, for curve k, along w_{i_k}, we only label the intersections of curve k and curves with higher curve numbers The intersections of curve k and curves with lower curve numbers must have been labeled already in some other w's

After we finish assigning vertex labels along $w_{i_{n-2}}$ for curve $n-2$, we obtain a vertex table of the intersections of curve i with curve j $(0 \leq i \leq j \leq n-1)$ We shall call this vertex table as a *initial table* The algorithm for constructing initial table is given in Figure 2 5

Step two In this step, For each w_{i_k} in the initial table, we assign vertex labels to the intersections of curve k with all curve c where $c < k$ We scan along string w_{i_k} $(k = 1, \ldots, n-1)$ starting at its first element $w_{i_k 0}$ until

CONSTRUCT_INITIAL_TABLE
Input $w_{\imath_0}, w_{\imath_1}, \ldots, w_{\imath_{n-1}}$

1 $v \leftarrow 0$
2 **for** $k = 0 \ldots n-2$
3 **for** each $w_{\imath_k \jmath}$ in $w_{\imath_k}$ for curve k
4 **if** ((the first component of) $w_{\imath_k \jmath} > k$)
5 associate $w_{\imath_k \jmath}$ with vertex v
6 $v \leftarrow v+1$
7 Output Initial Table $w_{\imath_0}, w_{\imath_1}, \ldots, w_{\imath_{n-1}}$ with vertex labels

Figure 2.5 Algorithm for Constructing Initial Vertex Table

we meet some $w_{\imath_k \jmath}$ $(0 \le \jmath \le m_{\imath_k})$ which has not associated with a vertex number in the initial table. Assume that the curve number $w_{\imath_k \jmath}$ be c (the first component of $w_{\imath_k \jmath}$ is c if it is written in a pair of a curve number and a segment number, or $c = w_{\imath_k \jmath}$ otherwise), then c must be less than $\imath_k$ and this is why $w_{\imath_k \jmath}$ is not associated with a vertex label in the initial table.

$$w_{\imath_k} \quad w_{\imath_k 0}(v_\star) \quad \cdots \quad w_{\imath_k, \jmath-1}(v_\star)\ w_{\imath_k, \jmath} \cdots$$

However, this intersection of curve k and curve c is also recorded in $w_{\imath_c}$, and a vertex label, say v_l, has been associated to the same intersection in $w_{\imath_c}$ in the initial table. We may assume the intersection in $w_{\imath_c}$ is $w_{\imath_c m}$ for some m.

$$w_{\imath_c} \quad w_{\imath_c 0}(v_\star) \quad \cdots \quad w_{\imath_c, m-1}(v_\star)\ w_{\imath_c, m}(v_l) \cdots$$

By the way we found out $w_{\imath_k \jmath}$, and the assumption on $\imath_k$'s, we have $m = P[c][k]$. (And this is exactly what P is for in G-encodings.)

$$w_{i_c} \quad w_{i_c 0}(v_\star) \quad w_{i_c,m-1}(v_\star)\ w_{i_c,m}(v_l)$$
$$\vdots \quad \vdots$$
$$w_{i_k} \quad w_{i_k 0}(v_\star) \quad w_{i_k,j-1}(v_\star)\ w_{i_k,j}$$

We then simply associate vertex v_l to $w_{i_k j}$

$$w_{i_c} \quad w_{i_c 0}(v_\star) \quad w_{i_c,m-1}(v_\star)\ w_{i_c,m}(v_l)$$
$$\vdots \quad \vdots$$
$$w_{i_k} \quad w_{i_k 0}(v_\star) \quad w_{i_k,j-1}(v_\star)\ w_{i_k,j}(v_l)$$

By Lemma 2 7, all other $w_{i_k,\star}$ for the same curve c in w_{i_k} can be associated vertex numbers deduced from w_{i_c} We then keep scanning the string w_{i_k} continuously until all intersections of curve k with any other curve (different) c $(c < k)$ are associated with vertex numbers After we finish scanning all w_{i_k} $(1 \leq k \leq n-1)$, we end up with a *completed vertex table* The algorithm for this step is shown in Figure 2 6

If two w strings in the given G-encoding are for the same curve, then the vertices on one w string is a circular shift of the other w strings The shifting number can be determined by counting the occurrence of each digit in w

We now have labeled all vertices along n curves Each vertex has a neighborhood consisting of four vertices (not necessarily different), two on each intersecting curve

Now we construct a circular list of four oriented edges for each vertex in the completed vertex table For simplicity, we assume curve c and curve k intersect at vertex v_1, and assume that in the complete vertex table we just built, we have (with $w_{i_c m}$'s omitted)

$$w_{i_c} \quad \cdots v_0\ v_1\ v_2 \cdots$$
$$w_{i_k} \quad \cdots v_3\ v_1\ v_4 \cdots$$

CONSTRUCT_COMPLETE_TABLE
Input Initial Table $w_{i_0}, w_{i_1}, \ldots, w_{i_{n-1}}$

```
1   for k = 1 ... n − 1
2     for each w_{i_k j} in w_k for curve k
3       c ← curve number of w_{i_k j}
4       if c < k and no vertex label associated with w_{i_k j}
5         m = P[c][j]
6         associate w_{i_k j} with the vertex associated with w_{i_c m}
7         N_k ← total vertex number on curve k
8         N_c ← total vertex number on curve c
9         s ← j + 1 mod N_k
10        t ← m + 1 mod N_c
11        while s ≠ j
12          while curve number of w_{i_k s} ≠ c, s = s + 1 mod N_k
13          while curve number of w_{i_c t} ≠ k, t = t + 1 mod N_c
14          associate w_{i_k s} with the vertex associated with w_{i_c t}
```

Figure 2.6 Algorithm for Constructing Complete Vertex Table

We can easily figure out that whether curve c is moving in or out of curve k at v_1 by counting the numbers of intersections between these two curves along the original w_{i_c} string of the G-encoding. If curve c meets curve k even (including 0) times up to v_1 (exclusive), then curve c is moving into the interior of curve k. This is important to build the circular list of oriented edges at vertex v_1. Without loss of the generality, we assume that curve c is moving into curve k at v_1. Therefore the circular list of v_1 is

$$\{(v_1, v_3)_k, (v_1, v_0)_c, (v_1, v_4)_k, (v_1, v_2)_c\},$$

where $(v_1, v_3)_k$ indicates the oriented edge from v_1 to v_3 along curve k. We can build a circular list of oriented edges in a similar way for each vertex in the completed vertex table.

The algorithm for building a circular list of the oriented edges for each vertex is given in Figure 2.7.

By Theorem 1.3, the given G-encoding determines a 2-cell embedding on some orientable surface S_h.

Assume that there are p vertices, q edges and r regions in the given diagram. We have p - q + r = 2 since the given diagram is planar. Actually it is not hard to show directly that $q = 2p$ and $r = p + 2$. When the diagram is 2-cell embedded into some surface S_h, by Euler's theorem (Theorem 1.2), we have p - q + r = 2 - 2h, so h is 0. This indicates the resulting pasting surface is a sphere.

By the definition of NFISC, there is a common interior to all curves. It is trivial to show that for a curve to contribute to the boundary of the common

CONSTRUCT_CIRCULAR_LIST
Input Completed Table w_{i_0}, w_{i_1}, ..., $w_{i_{n-1}}$

1 **for** each vertex v in the completed table
2 find the two curve numbers $c < k$ contain v
3 find the two vertices v_0, v_2 adjacent to v on curve c
4 swap v_0 and v_2 if necessary
so that v_0, v and v_2 is in clockwise order on curve c
5 find out the two vertices v_3, v_4 adjacent to v on curve k
6 swap v_3 and v_4 if necessary
so that v_3, v and v_4 is in clockwise order on curve k
7 count the number of times curve c and k meet before v on w_{i_c}
8 **if** even
9 $v \rightarrow \{(v, v_3)_k, (v, v_0)_c, (v, v_4)_k, (v, v_2)_c\}$
10 **else**
11 $v \rightarrow \{(v, v_3)_k, (v, v_0)_c, (v, v_4)_k, (v, v_2)_c\}$

Figure 2.7 Algorithm for Building Circular List of Oriented Edges

interior, the curve has to meet each of other curves odd times. By examing w_i's, it is not hard to find out which vertices touch the common interior, and hence what are the oriented edges may consist of the boundary for the 2-cell on the sphere which is corresponding to the common interior region of the desired diagram. Indeed, if some orbit of the mapping $\hat{\phi}$ consists entirely by the above oriented edges, the orbit contains the desired 2-cell. Similarly we can also find the oriented edges for the 2-cell on the sphere corresponding to the outer most face of the desired diagram. We then make a stereographic projection to obtain the plane n-diagram, by setting the north pole in the region on the sphere corresponds to the outermost face of the n-diagram, and the south pole in the region on the sphere which is to the common interior of the n-diagram. We are done. ∎

Example 2.10 This example gives a detailed illustration of Theorem 2.9 and Theorem 1.3. We show that how an NFISC can be determined given a G-encoding

$$\begin{array}{ccccc} 0 & 1 & 2 & [2] & [1] \\ 1 & 2 & [2] & 0 & [0] \\ 2 & 1 & 0 & [0] & [1] \\ 1 & 0 & [0] & 2 & [2] \end{array}$$

That is, $I = \{c_0, c_1, c_2, c_3\} = \{0,\ 1,\ 2,\ 1\}$, $w_0 = \{w_{00},\ w_{01},\ w_{02},\ w_{03}\}$, and $P[0][1] = 3$, $P[0][2] = 2$ and $P[1][2] = 1$.

We now start to label vertices along w_0, w_1, and w_2 which are corresponding to the first occurrences of curve 0, curve 1 and curve 2 in I respectively. Along curve 0, the intersections of curve 0 with curve 1 or curve 2 are labeled

consequently, and along curve 1, only the intersections of curve 1 with curve 2 are labeled We end up with the following table, which we call *initial table*

0	1(0)	2(1)	2(2)	1(3)
1	2(4)	2(5)	0	0
2	1	0	0	1
1	0	0	2	2

We then determine the vertex labels for the intersections corresponding to the rest of unlabeled w_{ij}s Start at $w_{12} = 0$ which is an intersection of curve 1 and curve 0 The vertex number for this intersection must have been assigned on curve 0 already And this vertex is the first one curve 1 meets curve 0 after curve 1 touches outer face, so this intersection is the one corresponding to w_{03} since $P[0][1] = 3$ (Lemma 2 7) Hence the vertex label for w_{12} is the same as the vertex label for w_{03}, which is vertex 3 Consequently, the vertex label for $w_{13} = 0$ is vertex 0 Similarly, we can determine all vertex labels along curve 2, and end up with the following table, which we call a *complete table*

0	1(0)	2(1)	2(2)	1(3)
1	2(4)	2(5)	0(3)	0(0)
2	1(5)	0(2)	0(1)	1(4)
1	0	0	2	2

Now for each vertex, we know which two curves it is on, and which vertices it is adjacent to For instance, vertex 0 is adjacent to vertex 3 and vertex 1 on curve 0, and vertex 0 is also adjacent to vertex 3 and vertex 4 on curve 1 To build a circular list of the oriented edges at vertex 0, we first count the number of times that curve 0 and curve 1 meet before vertex 0 along curve 0 This can be done by counting how many 1's in the given G-encoding We

have 0 (even) time that these two curves meet before vertex 0. We therefore know that curve 0 is moving into the interior of curve 1, and curve 1 is moving out from curve 0. So the circular list for vertex 0 is

$$0 \quad (0,3)_1, (0,3)_0, (0,4)_1, (0,2)_0$$

For another instance, let us exam vertex 2 which is on curve 0 and curve 2. Furthermore, from the above table, vertex 2 is adjacent to vertex 1 and vertex 3 on curve 0, and is adjacent to vertex 5 and vertex 1 on curve 2. In addition, we also know that curve 0 moves out curve 2 at vertex 2, because from the G-encoding string w_0, curve 0 and curve 2 intersect once (odd). With this in mind, we build the the circular list for vertex 2.

$$2 \quad (2,1)_0, (2,5)_2, (2,3)_0, (2,1)_2$$

The completed circular lists for each vertex are obtained similarly.

$$\begin{array}{ll} 0 & (0,3)_1, (0,3)_0, (0,4)_1, (0,2)_0 \\ 1 & (1,2)_2, (1,0)_0, (1,4)_2, (1,2)_0 \\ 2 & (2,1)_0, (2,5)_2, (2,3)_0, (2,1)_2 \\ 3 & (3,2)_0, (3,5)_1, (3,0)_0, (3,0)_1 \\ 4 & (4,1)_2, (4,0)_1, (4,5)_2, (4,5)_1 \\ 5 & (5,4)_1, (5,4)_2, (5,3)_1, (5,2)_2 \end{array}$$

We denote the set of the oriented edges as $E^\star = \{(i,j)_k\}$, and we define a map ϕ on $E^\star$ as we did in Theorem 1.3.

$$\begin{array}{rl} E^\star & \longrightarrow E^\star \\ (i,j)_k & \longrightarrow \text{the one after } (j,i)_k \end{array}$$

It is straitforward to show that the map is well-defined, one-to-one, and onto. For each edge $(i,j)_k$ in $E^\star$, $(i,j)_k$, $\phi((i,j)_k)$, $\phi^2((i,j)_k)$, , consist of a closed

walk In this way, we can obtain regions (2 cells) Then we paste the regions by gluing $(i,j)_k$ with $(j,i)_k$ together, we get an orientable surface (which is actually a sphere) on which the desired n-diagram is uniquely embedded

Furthermore, by studying w_0, curve 0 meet curve 1 once (odd) at vertex 0 and then curve 1 once (odd) at vertex 1, so vertex 1 must touch the common interior of all curves Hence the arc of curve 0 between vertex 1 and the next vertex (which is vertex 2 from w_0) must be a part of the boundary Similarly the arc of curve 2 between vertex 2 and vertex 1 (from w_2) is a part of the boundary of the common interior too

By examing all orbits of $(2,1)_0$, $(1,2)_0$, $(2,1)_2$ and $(1,2)_2$, we can find out which orbit is for the common interior for the desired diagram For instance, $\phi(1,2)_0$ is $(2,5)_2$ which is not on the boundary of the common interior anymore, so the orbit of $(1,2)_0$ does not contain the 2-cell which is corresponding to inner most face of the desired diagram However, $\phi(2,1)_0$ is $(1,2)_2$, and $\phi(1,2)_2$ is $(2,1)_0$, the original preimage, and the orbit is back to the starting preimage All elements of the orbit consist of the boundary of the common interior

It is straitforward, but tedious to show that one can determine the 2-cell region on the sphere which is corresponding to the outer most face of the desired diagram

We now can take a suitable stereographic projection to obtain a planar 3-diagram, which we showed that it is uniquely determined by the given G-encoding

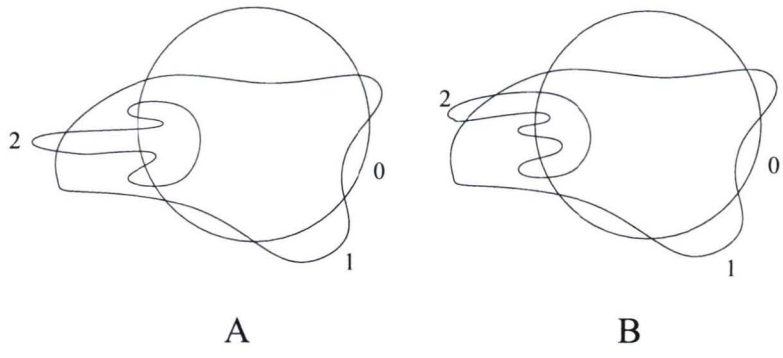


Figure 2.8 Matrices P for G-encodings

Example 2.11 This example explains why matrices P is necessary for the definition of G-encodings. Assume we are given the following G-encodings (without P), and we try to construct some 3-diagram.

$$\begin{array}{lll} 0 & w_0 = & 1\,1\,1\,2\,2\,2\,2\,2\,2\,1\,1\,1 \\ 1 & w_1 = & 0\,0\,2\,2\,0\,0\,0\,0 \\ 0 & w_2 = & 1\,2\,2\,2\,2\,2\,2\,1\,1\,1\,1\,1 \\ 1 & w_3 = & 2\,2\,0\,0\,0\,0\,0\,0 \\ 2 & w_4 = & 1\,0\,0\,0\,0\,0\,0\,1 \\ 1 & w_5 = & 0\,0\,0\,0\,0\,0\,2\,2 \\ 0 & w_6 = & 1\,1\,1\,1\,1\,2\,2\,2\,2\,2\,2\,1 \\ 1 & w_7 = & 0\,0\,0\,0\,2\,2\,0\,0 \end{array}$$

We can assign vertex labels along w_0 and w_1 according to the Algorithm in Figure 2.5, and obtain the following initial table.

$$\begin{array}{lll} 0 & w_0 = & 1(0)\ 1(1)\ 1(2)\ 2(3)\ 2(4)\ 2(5)\ 2(6)\ 2(7)\ 2(8)\ 1(9)\ 1(10)\ 1(11) \\ 1 & w_1 = & 0\ 0\ 2(12)\ 2(13)\ 0\ 0\ 0\ 0 \\ 2 & w_4 = & 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1 \end{array}$$

To obtain the complete vertex tables, we need to determine what the vertices are for the intersections on curve 2 with curve 0, and with curve 1. However, we can not uniquely determine for all the intersections of curve

2 and curve 0 in this example, due to lack of information For example, for w_{41} in string w_4, where (segment 0 of) curve 2 meets curve 0 first time, all we know is that w_{41}, the first 0 in w_4 should be corresponding to some vertex associated with $w_{0m} = 2$ in w_0 where $m = 3, 4, \ldots, 8$ Diagram A and B in Figure 2 8 have the same G-encodings as above For Diagram A, the intersection denoted by w_{41} is the same one denoted by w_{06} But for Diagram B, it is w_{08} corresponding to w_{41} The matrix P in the definition of G-encoding is for solving the undetermined problem For instance, in Diagram A, $P[0][2] = 6$ should be given, indicating that w_{06} is the same intersection of the segment 0 of curve 2 with curve 0 Similarly in Diagram B, $P[0][2] = 8$ should be given, indicating that w_{08} is the same intersection of the segment 0 of curve 2 with curve 0

2 1.2 A Special Case: Grünbaum Encoding

The Grunbaum encoding of a simple symmetric monotone (SSM) Venn diagram consists of four n-ary strings, which we call w, x, y, z To obtain string w for the diagram, we first order the curves by their clockwise appearance on the outer most region so that all curves are labeled with $0, 1, \ldots, n-1$ In order to obtain string w, we follow curve 0 in a clockwise direction, starting at a point where it touches the outer most region and meets curve 1, recording its intersections with the other curves, until we reach the starting point again

String x is obtained in a similar manner, by following curve 0 in a clockwise direction, starting at the point where it touches inner most region and crosses with another curve (which must actually be curve 1), and recording its intersections with the other curves

To obtain the third string y, we must re-label the curves counter-clockwise as they appear on the outer most region We follow curve 0 in a direction, starting at the point where curve 0 is out most and is intersected by curve 1, and record curve 0's intersections with other curves

String z is obtained by following curve 0 in a counter-clockwise direction, starting at the point where it is most inside and crosses with some curve (it must be curve 1 again), recording its intersections with the other curves

Example 2 12 The Venn diagram Victoria (Figure 2 9) has the following Grünbaum encoding

w 1 4 2 5 3 6 1 6 3 5 3 6 2 5 1 6 1 5 3 6 2 5 1 4 2 6 1 6 2 5 1 4 2 4 1 6

x 1 6 3 5 3 6 2 5 1 6 1 5 3 6 2 5 1 4 2 6 1 6 2 5 1 4 2 4 1 6 1 4 2 5 3 6

y 1 6 3 5 3 6 2 5 1 6 1 5 3 6 2 5 1 4 2 6 1 6 2 5 1 4 2 4 1 6 1 4 2 5 3 6

z 1 4 2 5 3 6 1 6 3 5 3 6 2 5 1 6 1 5 3 6 2 5 1 4 2 6 1 6 2 5 1 4 2 4 1 6

Example 2 13 The Venn diagram M1 (Figure 2 10) has the following Grunbaum encoding

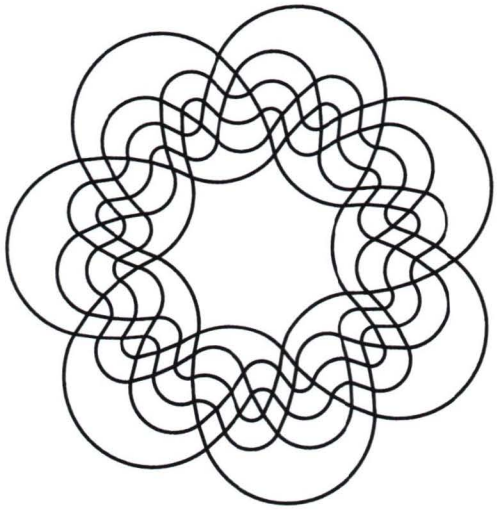


Figure 2.9 Victoria

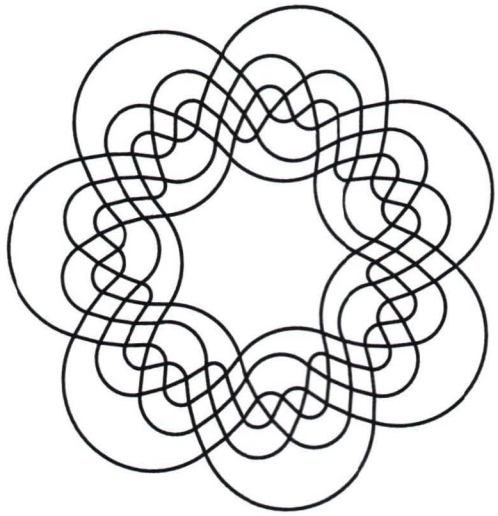


Figure 2.10 M1

w 1 4 2 6 2 4 2 5 3 6 1 3 5 3 6 4 1 6 2 1 3 2 3 5 1 6 1 4 3 5 4 5 2 5 4 6

x 1 4 3 5 4 5 2 5 4 6 1 4 2 6 2 4 2 5 3 6 1 3 5 3 6 4 1 6 2 1 3 2 3 5 1 6

y 1 3 2 5 2 3 2 4 3 6 1 6 2 4 5 4 6 5 1 6 3 1 4 2 4 6 1 4 2 5 3 5 1 5 3 6

z 1 6 2 4 5 4 6 5 1 6 3 1 4 2 4 6 1 4 2 5 3 5 1 5 3 6 1 3 2 5 2 3 2 4 3 6

Although any one of the four strings of a Grünbaum encoding can produce another three strings, we will still calculate and have all four strings for each Venn diagram. Indeed, any Venn diagram can generate another three isomorphic Venn diagrams by "flipping" and/or "polar flipping" mappings. The strings x, y and z are the first string of the Grünbaum encoding of these isomorphic diagrams respectively. Keeping all four strings for each Grünbaum encoding will be convenient to verify isomorphisms of Venn diagrams.

Property 2.14 *Each string of the Grünbaum encoding of a simple symmetric monotone n-Venn diagram has length* $(2^{n+1} - 4)/n$.

Proof Let m be the length of the string w, which records the intersections of curve 0 with other curves. If we construct such a string for each curve then there are n such strings to record all intersections. Since the digram is simple, each intersection is counted twice. These strings must have the same length m, since the diagram is symmetric. There are altogether 2^n regions in the Venn diagram, among them one is the most outside region and one is the most inside region. Therefore there are $2^n - 2$ intersections altogether. Hence we have $nm = 2(2^n - 2)$, so $m = (2^{n+1} - 4)/n$. ∎

Property 2.15 *Each string starts with 1 ends with $n - 1$.*

Proof This property is obvious given the definition of Grünbaum encoding ∎

Property 2.16 *String x, y and z can be inferred from string w*

Proof Let L denote the length $(2^{n+1}-4)/n$ of the Grünbaum encoding, and let $w[i]$ and $y[i]$ be the ith bit of w and y respectively, where $0 \leq i \leq L-1$. Then

$$y[i] = n - w[L-i-1] + 1$$

To obtain x, we first find out the unique location where all curves have been encountered an odd number of times, then shift w circularly at this location. The string z can be inferred from y as x from w. ∎

In Example 2.12, the string x, y and z can be inferred from w as follows: to obtain x, we simply shift w at the position where the second "1" occurs, since before this position, each number occurs one time, to obtain z from y we shift y at the position where the eighth "1" occurs since before the position each number occurs odd number of times.

Corollary 2.17 *If an n-Venn Diagram is polar symmetric then its Grünbaum encoding consists of two pairs of identical strings.*

Proof Since the given Venn diagram is polar symmetric, by the way of its Grünbaum encoding being constructed, we have $w = z$ and $x = y$. ∎

The following theorem is a direct result of Theorem 2.9.

Theorem 2.18 *Each Grünbaum encoding determines a unique (up to isomorphisms) simple symmetric monotone n-Venn diagram.*

Proof It suffices to show that Grünbaum encodings are actually a G-encodings for SSM Venn diagrams. Given a Grünbaum encoding, we use the labels of the curves of Grünbaum encoding as the ones for G-encoding. Therefore, $I = \{0, 1, 2, \ldots, n-1\}$. Each curve has exactly two segments, since each curve touches outer face exactly once. Moreover, we have $w_0 = x$ of Grünbaum encoding, (and there are no need to write w_0 as a string of pairs of integers). Other w_i can be deduced from w_0. $w_{ij} = w_{0j} + i \bmod n$.

To determine the value of $P[i][j]$ with $0 \leq i \leq j \leq n-1$, we first find out the first k such that $w_{jk} = i$, and we count the even or odd encounter times of curve j with each of curve 0, 1, $\ldots$, $n-1$ respectively (odd for j) up to w_{jk}. Then in string w_i, we look for some m such that $w_{im} = j$ and before w_{im} the even or odd occurrences of 0, 1, $\ldots$, $n-1$ (except i) are the same as we just calculated for w_{jk}. The properties of Venn diagrams guarantees that there exits exactly one such m. We let $P[i][j] = m$.

In this way, we can construct a G-encoding. This G-encoding only depends on which string of x, y, z or w is chosen to be the first string w_0 of G-encoding. Since there exist permutation mappings for one to deduce all other strings from any one of x, y, z or w. It is not hard to show that all the possible G-encodings which are deduced from the given Grünbaum encoding are *equivalent*, and hence they determine a same SSM n-Venn diagram. Thus the Grünbaum encoding uniquely identifies this SSM n-Venn diagram. ∎

2.2 The Matrix Representations

Simple symmetric monotone Venn diagrams can be represented by matrices. Because of symmetry, it is sufficient to represent one sector with radian $2\pi/n$ for each diagram. One can always rotate the sector to generate the whole diagram.

Given a sector of an n-Venn diagram, one can map this piece of sector into a graph consisting of crossing polygonal curves (poly-lines). We then put 0 between these $n-1$ poly-lines and put a 1 at each crossing point. In this way a 0-1 matrix is generated. We shall call the matrix a *representation matrix* for the Venn diagram. Example 2.19 and Example 2.20 show how the representation matrices for the Venn diagram Victoria and M1 are generated.

Example 2.19 In Figure 2.11, one sector of Venn diagram Victoria is represented by some line segments, which we call *poly-lines*. In Figure 2.12, 0's and 1's are inserted into the sectors. After removing the poly-lines, we obtain the matrix representation of Venn diagram Victoria.

Example 2.20 Figure 2.14, 2.15 and 2.16 show how the matrix representation for Venn diagram M1 is obtained.

On the other hand, from a representation matrix we can easily draw suitable poly-lines to construct the corresponding sector of the Venn diagram. We can then expand the matrix by appending identical matrix blocks to generate a matrix which represents the whole Venn diagram.

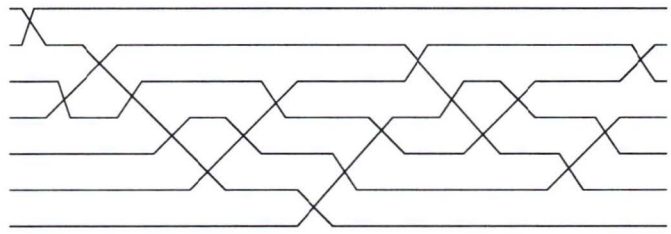


Figure 2.11 The Poly-line of Victoria

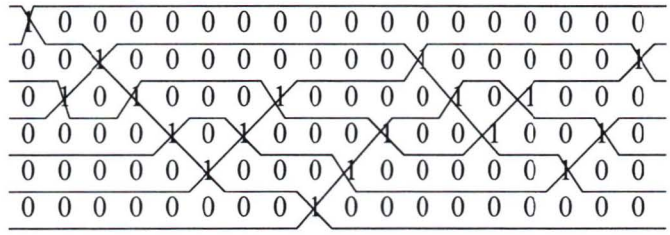


Figure 2.12 The Matrix and the Poly-line of Victoria

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
0	1	0	1	0	0	0	1	0	0	0	0	1	0	1	0	0	0
0	0	0	0	1	0	1	0	0	0	1	0	0	1	0	0	1	0
0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Figure 2.13 The Matrix of Victoria

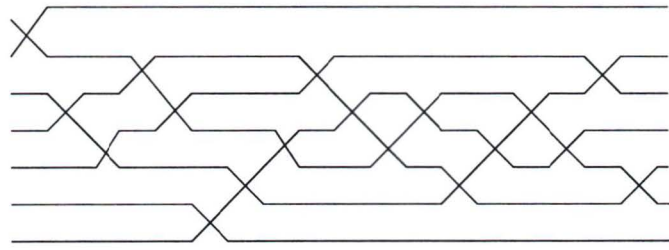


Figure 2.14 The Poly-line of M1

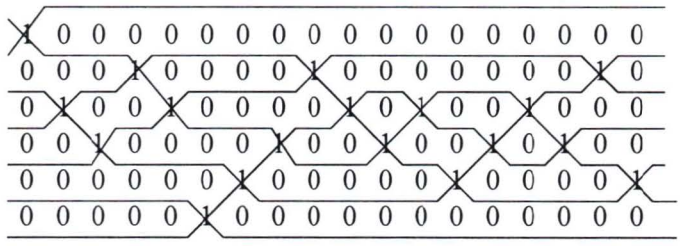


Figure 2.15 The Matrix and the Poly-line of M1

1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0
0	1	0	0	1	0	0	0	0	1	0	1	0	0	1	0	0	0
0	0	1	0	0	0	0	1	0	0	1	0	0	1	0	1	0	0
0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2.16 The Matrix of M1

Intuitively, we can generate SSM Venn diagrams by generating suitable 0-1 matrices. However, the problem is that computational complexity would be too high if there were no restrictions on the matrices. The following theorem gives a necessary condition for representation matrices of SSM Venn diagrams.

Theorem 2.21 *For the representation matrix of one sector of an n-Venn diagram, there are $(2^n - 2)/n$ 1's in each matrix. There are $\binom{n}{k}/n$ 1's in the kth row, where $k = 1, 2, \ldots, n-1$.*

Proof For a SSM n-Venn diagram there are $2^n - 2$ interior regions altogether. They are evenly distributed in each sector. Hence in each sector there are $(2^n - 2)/n$ regions.

For representing such a region with a matrix, we use a 1 to indicate the starting point of a region and the ending point of the next adjacent region. This implies that there are exactly $(2^n - 2)/n$ 1's in the matrix.

For each $k = 1, 2, \ldots, n-1$, the number of the regions which are inside k curves and outside of other $n-k$ curves is obviously $\binom{n}{k}$ for the whole SSM n-Venn diagrams. Therefore, in each sector, there are $\binom{n}{k}/n$ such regions, it turns out that there are $\binom{n}{k}/n$ 1's in the kth row of a representation matrix.

■

Remark 2.22 *[5] [12] For an integer n such that n is a divisor for all $\binom{n}{k}$, $k = 1, 2, \ldots, n-1$, n is necessarily prime.*

It thus follows that, for $n = 7$, we have the following corollary.

Corollary 2.23 *For the representation matrix of one sector of an 7-Venn diagram, there are eighteen 1's in each matrix. There is one 1 in the first row and the last row, three 1's in the 2nd row and fifth row, five 1's in the third row and fourth row.*

By the above theorem, with exactly one 1 in each column, one may always use an $(n-1) \times (2^n - 2)/n$ 0-1 matrix to represent the sector and use an $(n-1) \times (2^n - 2)$ 0-1 matrix to represent the whole diagram. In this paper, we always use 6×18 matrices for one sector of an SSM 7-Venn diagram.

The following theorem reduces the computational complexity by adding more restrictions on the representation matrices of SSM Venn diagrams.

diagrams? To solve this problem the Grunbaum encoding technique is employed.

2.3 Graph Representations

2.3.1 The Definition

The graph representation of a SSM n-Venn diagram is an "adjacency-list representation" [3]. Given an SSM n-Venn diagram V, we can label all of the vertices of V 0, 1, , up to $2^n - 2$. The graph representation consists of an array a of $2^n - 2$ circular lists, one for each vertex in V. For each vertex labeled by an integer k, $a[k]$ is a circular list of the labels of the neighbors of vertex k, in clockwise direction. A graph representation a of an SSM n-Venn diagram V is related to the labeling of V's vertices. Therefore an SSM Venn diagram can have $(2^n - 2)!$ many graph representations. For each graph representation of an SSM n-Venn diagram, we have

Theorem 2.25 *The graph representation of an SSM n-Venn diagram is an array a of $2^n - 2$ circular lists, with each list having 4 elements.*

Proof We have shown that there are $2^n - 2$ vertices in V. Since V is simple, each vertex u is an intersection of two Jordan curves. Therefore u's neighbor consists of 4 vertices, two on each curve. ∎

2.3.2 Graph Representations and Isomorphisms

It is well-known that, if two n-Venn diagrams have the same graph representation, then these two n-Venn diagram must be isomorphic.

Given two SSM n-Venn diagrams, each with a graph representation, our interest here is to determine whether they are isomorphic One way to do this is to relabel all the vertices of one Venn diagram in all possible ways (there are $(2^n - 2)!$ altogether), get a new graph representation each time, and compare it with the graph representation of the other Venn diagram If one representation of the first Venn diagram matches up with the other's, then these two diagrams are isomorphic Fortunately, since the given Venn diagrams are simple, symmetric, and monotone, we only need to relabel them a few times each each time we choose some special starting vertex, to determine whether they are isomorphic or not

2.3.3 Choosing Suitable Graph Representations

When we need to determine whether two SSM n-Venn diagrams are isomorphic, we do not need to compare all of their graph representations, we can achieve the same result by comparing some special graph representations of these two diagrams Since graph representations are related to the labeling of vertices, first problem is how to label the vertices to obtain the special graph representations for each diagram

The vertex relabeling method is induced from the *Depth-first Search*, or DFS [3] and involves doing a depth-first search on a given SSM Venn diagrams V, and relabel the vertices in the order they are visited There are two versions of DFS for our purpose the leftmost DFS and the rightmost DFS During the left DFS process, we imagine that we are walking on the

edges (or curves) of the diagram If we enter a vertex, we choose one of the neighboring vertices to go to, starting from our *left* side The vertex we move to will be the first unvisited vertex on our left side This makes sense since we are on an orientable surface and the neighbor is represented by a circular list, in clockwise order We can call this search the *leftmost version of depth-first search* The rightmost version of depth-first search can be described in a similar manner Obviously, doing a left DFS on a SSM Venn diagram V is equivalent to doing a rightmost DFS on the mirror image of V After all the vertices of V are relabeled, we obtain a new graph representation by constructing (or updating) its vertex-adjacent list

Assume we are given two SSM 7-Venn diagram V_1 and V_2 with labeled vertices and graph representations, and we need to determine whether they are isomorphic To do leftmost and rightmost DFS searches for each SSM n-Venn diagram V, we can only choose a vertex u which touches the most outside region and a vertex v which touches the most inside region as the starting points For the starting points, we need an agreement as to which vertex is on our left and which is on our right side, since we may turn around at the starting points For instance, assume the vertex u we chose above has a neighbor (circular list in clockwise) $\{w_0,\ w_1,\ w_2,\ w_3\}$ If we assume w_i is on our left, then w_{i+1} is to our front, w_{i+2} to our right, and w_{i+3} behind us (using modular addition if necessary) Notice that since u touches the outermost region, two of w's must do so as well To reduce the computational time, we will only choose one of these two w's as it is behind us when we start to do

leftmost or rightmost DFS. Therefore, starting from u, we will do two leftmost DFS and two rightmost DFS. In each case, we always know which will be the next vertex to visit from the starting vertex u. Similarly, we do two leftmost and two rightmost DFS starting from v, for a total of 8 DFS's for each given SSM 7-Venn diagram. We can relabel the vertices after each search, and accordingly get an updated graph representation each time. The 8 typical graph representations so obtained are called *normal graph representations*. From the discussion above, we have the following theorem which provides us with another method to identify a SSM n-Venn diagram.

Theorem 2.26 *Any two isomorphic SSM n-Venn diagrams have the same 8 normal graph representations.*

The detailed algorithm for graph representation and searching all SSM 7-Venn diagram will be presented in Chapter 4.

Chapter 3

The Algorithm Using Grünbaum Encoding

Our first algorithm to search all simple symmetric monotone (SSM) 7-Venn diagrams is given in (Figure 3.1). In this algorithm, we use Grünbaum encoding to distinguish Venn diagrams. We use the set S to hold the SSM 7-Venn diagrams obtained. For each diagram V, we use g_V to denote its Grünbaum encoding.

3.1 Generating Matrices

According to Theorem 2.21 and Corollary 2.23, for the case $n = 7$, each 6×18 matrix M generated in step 2 has 18 entries with value 1, and the other 90 entries are valued 0. The 18 entries with value 1 are distributed with the following restriction.

- 1 entry in each column,
- 1 entry in row 0 and row 5 respectively,

COMPUTE_ALL_1

```
1   S ← Nil
2   for each of all possible 6 × 18 0-1 representing matrices
3     Expand the matrix to 6 × 126 matrix M
4     Construct the poly-lines
5     if the poly-lines consist of a Venn Diagram V
6       calculate Grünbaum encoding g_V for V
7       compare g_V with g_{V_i} of each V_i in S
8       if g_V ≠ g_{V_i} for all i
9         V is a new solution, add V into S
10      goto step 2
```

Figure 3.1 Algorithm One

- 3 entries in row 1 and row 4 respectively,
- 5 entries in row 2 and row 3 respectively,
- the entry in row 0 is also in column 0 (i.e. $M[0][0] = 1$),
- no adjacent entries (with value 1) in any row.

Initially, M is set as a zero matrix, and then the entries with value 1 can be simply selected with 17 *for* loops. To reduce the computational time, we can use two additional matrices F and A, in order to guarantee that there are the desired number of entries with value 1 in certain row, and that there are no adjacent entries with value 1 in any row.

The first matrix F is 1×18. The value $F[j]$ $(0 \leq j \leq 17)$ indicates whether the corresponding column i has an entry with value 1.

$$\begin{aligned} F[j] &= 1 \quad \text{if for some } i \text{ such that } M[i][j] = 1, \\ &= 0 \quad \text{otherwise} \end{aligned}$$

The second matrix A is a 6×18 matrix. Its purpose is to indicate the available entries which can be assigned to 1 in a given row. These two matrices have to be updated every time a row has been successfully constructed. The following example illustrates how a representing matrix is generated and the usage of matrices F and A.

Example 3.1 Suppose the following matrix M is in the generating process, the first three rows (row 0 to row 2) have been successfully generated, and we are in position to select five entries in row 3 and assign value 1 to the entries.

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1

0 1 0 1 0 0 0 1 0 0 0 0 1 0 1 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

The matrix F is

1 1 1 1 0 0 0 1 0 0 0 1 1 0 1 0 0 1

The matrix $A[3][]$ for row 3 (the fourth row) has nine valid entries (only nine entries remained to be selected)

4 5 6 8 9 10 13 15 16

The other entries $A[3][9]$, $A[3][10]$, . . . , $A[3][17]$ could be simply assigned to -1. In this case the five entries in this row will be chosen from column 4, 5, 6, 8, 9 10, 13, 15, and 16. The only restriction is that no consecutive columns be selected. For instance, the choices of the five entries could be

{4, 6, 8, 10, 13},

{4, 6, 8, 10, 15},

{4, 6, 10, 13, 16},

. . .,

In general for row 3, we can select five entries by using five *for* loops. Since in this row there always nine columns available if row 0, row 1, and row 2 have been finished, the first chosen entry has to be from the columns which are in the first five available columns as well as in the first 10 columns. In other words, the entry must be chosen from the first five available columns with a column value of no more than 10 in order to leave enough choices for the other four entries. The second entry to be selected has be within the first six available entries for this row with a column value no greater than 11, and at least two column away from the entry already chosen, so that therefore no consecutive 1's in this row. Similar rules apply to the next three entries to be selected. The initial values for the *for* loops may be varied, however, this can be easily done with one or at most two comparison operations.

3.2 Constructing Poly-lines

For each 6×18 matrix M generated in Step 2 of Algorithm One, we must first expand it to a 6×126 matrix by appending the original matrix by itself six times. We still denote the expanded matrix by M. The behavior of 7 poly-lines can be inferred from matrix M: two poly-lines intersect with each other at some point if the corresponding entry of M is 1.

We can use a 7×126 *poly-line matrix* L to represent the poly-lines inferred from M. The first column of L is $[0, 1, 2, 3, 4, 5, 6]^T$. The rest of the entries will be evaluated with the Algorithm in Figure 3.2.

Example 3.2 The poly-line matrix for Victoria (the first 18 columns)

```
UPDATE_POLY_LINES
1   for i = 0 to 6
2       L[i][0] ← i
3   for r = 0 to 5
4       for c = 1 to 124
5           if M[r][c] = 0
6               L[r][c+1] ← L[r][c]
7           else L[r][c+1] = L[r+1][c]
8               L[r+1][c+1] ← L[r][c]
9               r ← r+1
10          if M[5][c] = 0, L[6][c+1] ← L[6][c]
```

Figure 3.2 Algorithm for Constructing a Poly-line Matrix

0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 0 3 3 3 3 3 3 3 3 3 5 5 5 5 5 5 5
2 2 3 0 2 2 2 2 5 5 5 5 3 6 6 2 2 2 2
3 3 2 2 0 4 4 5 2 2 2 6 6 3 2 6 6 4 4
4 4 4 4 4 0 5 4 4 4 6 2 2 2 3 3 4 6 6
5 5 5 5 5 5 0 0 0 6 4 4 4 4 4 4 3 3 3
6 6 6 6 6 6 6 6 6 0 0 0 0 0 0 0 0 0 0

Example 3.3 The poly-line matrix for M1 (the first 18 columns)

0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 0 0 3 3 3 3 3 4 4 4 4 4 4 4 4 4 2
2 2 3 3 0 4 4 4 4 3 6 6 0 0 0 2 2 4 4
3 3 2 4 4 0 0 0 6 6 3 0 6 6 2 0 6 6 6
4 4 4 2 2 2 2 6 0 0 0 3 3 2 6 6 0 0 0
5 5 5 5 5 5 6 2 2 2 2 2 2 3 3 3 3 3 3
6 6 6 6 6 6 5 5 5 5 5 5 5 5 5 5 5 5 5

3.3 Validation

A matrix M represents a Venn diagram if and only if the represented diagram has $2^7 - 2 = 126$ distinct regions. To distinguish all the regions from each

other we can label these regions with ranks. The *rank* of a given region is defined by

$$\text{rank} = 2^0 x_0 + 2^1 x_1 + \cdots + 2^{n-1} x_{n-1},$$

where $x_i = 1$ if the region is outside of poly-line i and $x_i = 0$ otherwise. By this definition, we therefore have

Theorem 3.4 *The upper bound of the ranks of the regions in an n-Venn diagram is* $2^n - 1$, *which is for the outer most face. And the lower bound of the ranks is 0, which is for the inner most face. For* $n = 7$, *the upper bound is 127.*

Corollary 3.5 *A matrix M represents a Venn diagram if and only it has exactly 126 regions with different ranks valued from 1 to 126.*

For each representing matrix M, a unique poly-line matrix L has been constructed in the previous step. To see whether the represented diagram is a Venn diagram, we can scan the M column by column from left to right. When an entry with value 1 is found, one region has just scanned successfully and another region is ready to be scanned. The rank of the scanned region can be calculated according to the entries in the same column of the poly-line matrix L. That is, suppose we scan M up to its column j and $M[i][j] = 1$ for some i $(0 \leq i \leq 5)$, then the rank of the region we just scanned is

$$2^0 L[0][j] + 2^1 L[1][j] + \cdots + 2^6 L[6][j].$$

All regions can be stored in a 1×128 matrix R, indexed by the ranks of regions (Theorem 3.4).

With ranks, we can identify the regions and therefore immediately know that the diagram represented by M is not a Venn diagram if some regions scanned more than once. At this point we can terminate the scanning process for the current matrix M.

If the scanning process is finished successfully, we can then check the regions of the diagram to see whether there are exactly 126 different regions presented in the diagram (except the most inside one and the most outside one). The algorithm for testing whether a generated matrix M represents an SSM Venn diagram is given in Figure 3.3. In this algorithm, M is the given 6×126 matrix we generated in previous steps, L is the poly-line matrix associated with M, and R is a one-dimensional matrix, each indexed by *ranks* of regions. For each i, $0 \leq i \leq 127$, $R[i]$ is the number of times the region with rank i is scanned. An integer variable k is for the number of regions scanned. The algorithm will return true if the diagram represented by a given matrix M is a Venn diagram. Otherwise, it returns false.

The idea of the algorithm is this: during the process of scanning M from column 0 to column 125, we are looking for the entry with value 1 in each column. Without loss of generality, assume we see $M[i][j] = 1$ for some nonnegative $i \leq 5$ and $j \leq 125$. We know a region has just scanned and another region is to be scanned. The ranks of these two regions can be calculated with the values of entries in jth column and $j + 1$th column in

IS_VENN_DIAGRAM

1 count $k \leftarrow 0$,

2 **for** column $j = 0$ **to** 125

3 **for** row i such that $M[i][j] = 1$

4 $k \leftarrow k + 1$

5 $r \leftarrow 2^{L[0][j]} + 2^{L[1][j]} + \quad + 2^{L[i][j]}$

6 $R[r] \leftarrow R[r] + 1$

7 **if** $R[r] > 1$, **return** false

8 return true

Figure 3 3 Algorithm for Validation of Venn Diagrams

matrix L (mod 126 if necessary)

If we do find a Venn diagram at this stage, we are ready to calculate its Grünbaum encoding

3.4 Generating Grünbaum Codes

As was shown in Section 3 2, 7 poly-lines can be constructed according to each matrix M generated The poly-lines are labeled in the up down order in the first sector of the diagram represented by M, and the behavior of the poly-lines is recorded in L To generate Grünbaum encoding g, we first have to relabel the poly-lines by order of appearance in the first row so that all poly-lines are labeled with 0, 1, , 6, since this is how Grünbaum codes are defined We will still use L for the relabeled poly-lines

The Grünbaum encoding g for the SSM 7-Venn diagram represented by M consists of 4 sequences w, x, y and z. Each is a sequences of 0, 1, , and 6 (Section 2.1.2). In the following section, we will show an algorithm for obtaining w, and will infer x, y and z from it.

Example 3.6 The relabeled poly-line matrix L for Victoria (the first 18 columns):

0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 0 4 4 4 4 4 4 4 4 4 3 3 3 3 3 3
2 2 4 0 2 2 2 2 3 3 3 3 4 6 6 2 2 2
4 4 2 2 0 5 5 3 2 2 2 6 6 4 2 6 6 5
5 5 5 5 5 0 3 5 5 5 6 2 2 2 4 4 5 6
3 3 3 3 3 3 0 0 0 6 5 5 5 5 5 5 4 4
6 6 6 6 6 6 6 6 6 0 0 0 0 0 0 0 0 0

The sequence w can be obtained from L directly. It is the sequence of the labels of the poly-lines which the poly-line 0 meets when it travels from column 0 to column 125 in matrix L. For example, we can obtain the first several digits of w from Example 3.6:

w : 142536 . . .

Figure 3.4 shows the algorithm for obtaining w and x given matrix L. The variable p is for the index of the sequence w, s is the number for circular

GENERATING_GRUNBAUM_ENCODING

```
1   p ← 0, s ← 0, f ← false,
2   for column j = 0 to 125
3     for row i such that M[i][j] = 1
4       if L[i][j] = 0, w[p] ← L[i + 1][j], p ← p + 1
5         if i = 5 and f = false, f ← true
6         else if f = false, s = s + 1
7       else if L[i + 1][j] = 0, w[p] ← L[i][j], p ← p + 1
8         if f = false, s ← s + 1
9   for i = 0 to p
10    x[i] = w[(i + s) mod p]
```

Figure 3.4 Algorithm for Generating Grünbaum Encoding

shifting on w. After w is obtained, x can be obtained by circular shifting w by the number s.

If we relabeled the poly-line in the reverse order of their appearance in the first row of the poly-line matrix, the first row of L will be changed from

$$0 \quad 0\ 1 \quad 1 \quad 6 \quad 6 \longrightarrow 6 \quad 6\ 5 \quad 5 \quad 0 \quad 0$$

This actually defines a permutation Π on $\{0, 1, \ldots, 6\}$. By applying this permutation to the sequences of w and x, and reversing the resulting sequences, we obtain sequences y and z. The desired Grünbaum encoding for the Venn diagram represented by M is $g = \{w, x, y, z\}$.

Each time we obtain a Grünbaum codes g for a generated matrix M, we compare g with the Grunbaum code of all the Venn diagrams we have already found (stored in set S). If g is equivalent to some Grünbaum encoding for some Venn diagram V in the set S of the solutions, then the Venn diagram M represents is isomorphic to the Venn diagram $V \in S$. In this case, matrix M and its Grünbaum encoding will not be added into set S again, otherwise, we add this M and g into S.

Chapter 4

The Algorithm Using Graph Representations

Our second algorithm to search all SSM 7-Venn diagrams is given in (Figure 4.1). Since the diagrams represented by generated matrices M are symmetric, it is sufficient for us to calculate only 8 *normal* graph representations for each diagram in order to identify it (Section 2.3.3).

The methods for generating 0-1 matrix M and poly-line matrix L are exactly the same as in the previous chapter. In this chapter, we will only discuss the methods for constructing graph representations. As we pointed out in Section 2.3, for each diagram V represented by M, we will

- label the vertices, and construct a graph representation for V;
- choose two vertices u and v, which touch the most outside region or most inside region respectively;
- do two leftmost and two rightmost depth-first searching on u and v;

```
COMPUTE_ALL_2
1   S ← Nil
2   for each of all possible 6 × 18 0-1 representing matrices
3       Expand the matrix to 6 × 126 matrix M
4       Construct the poly-lines L
5       if the poly-lines consist of a Venn Diagram V
6           calculate the 8 normal graph representations G_V[i]_{i=0}^{i=7} for V
7           compare G[0] with G_{V_i}[0] of each V_i in S
8           if V is a new solution, add V into S
```

Figure 4.1 Algorithm Two

- relabeling the vertices of v after each search,
- construct a (*normal*) graph representation after each search,
- sort these 8 normal graph representation in lexicographic order,

The normal graph representations of each diagram are sorted in lexicographic order, denoted by $G[0]$, $G[1]$, , $G[7]$. When we determine whether two given SSM 7-Venn diagrams are isomorphic, we only need to compare the first ones ($G[0]$'s) of the 8 normal graph representations. In fact, we only need to store the $G[0]$ as a unique graph representation.

4.1 Constructing Graph Representations

For each generated 0-1 6×126 matrix M and the poly-line matrix L associated with it, we can label all vertices of the diagram V represented by M as 0, 1, , 125. A graph representation can be constructed according to these vertex labels, these labels and the graph representation will be referred as the "original labels" and the "original graph representation." The upcoming depth-first searching and vertex relabeling will work on the diagram with original labels.

The vertices are labeled row by row, and column by column on each row. We can use a new 6×126 matrix N to record the labels of the vertices. The entries of N with value 0 are in the exactly the same position as those in Matrix M. The entries with value 1 in M are replaced in N by the label value of the corresponding vertices. Then, for each vertex i, we check its neighbors and construct an adjacent list for it, in clockwise order. The algorithm is in Figure 4.2.

4.2 Leftmost and Rightmost Depth-first Searching

After all the vertices of the diagram V are represented by matrix M generated in Algorithm Two, we choose two vertices, which touch the most outside region or most inside region respectively. We then perform four depth-first searching: two leftmost and two rightmost depth-first searching (DFS) on

```
LABELING_VERTICES
1   k ← 0
2   for row i = 0 to 5
3       for column j = 0 to 125
4           if M[i][j] = 1, N[i][j] ← k  k ← k + 1
5           else N[i][j] ← 0
```

Figure 4.2 Algorithm for Labeling Vertices

```
CONSTRUCT_GRAPH_REPRESENTATION
1   color all vertices WHITE
2   for row i = 0 to 5
3       for j = 0 to 125
4           if M[i][j] = 1
5               k ← N[i][j]
6               g[k] ← k
7               search in poly-line matrix L to find neighbor of vertex k
8               find out the labels of the neighbor of k in matrix N
9               make an adjacent list for vertex k
```

Figure 4.3 Algorithm for Constructing Graph Representations

both of them

During the process of a DFS, at each vertex u, we know which vertex in u's neighborhood is the leftmost unvisited vertex to u, since we know which vertex w is behind us, i e , from which vertex w (in neighbor of u) we come from However, for the starting point of each search, we have to define what is behind us, so that the left and right side make sense Since for the starting point which touchs the most outside region, there are two curves lead into the inside of the diagram, and the left and right side vertex of the starting point is determined by the way how we walk into the diagram, it is necessary to do two leftmost DFS

The algorithms in Figure 4 4, Figure 4 2, and Figure 4 6 are for the leftmost and rightmost DFS

After each search, all vertices in diagram V represented by matrix M will be relabeled The mapping of the old labels and the new labels is represented as an array F in algorithm DFS (Figure 4 6) We thus find a permutation on the set $\{0, 1, \quad , 125\}$, and can then update matrix N (which records the labels of the vertices), with this permutation mapping

$$N[i][j] \leftarrow F[N[i][j]]$$

A new (or *normal*) graph representation can be obtained according to the new labels of the vertices, this is accomplished in the algorithm in Figure 4 4 There are 8 normal graph representations obtained altogether and they are sorted in lexicographic order

OBTAIN_NORMAL_GRAPH_REPRESENTATION

Input: vertex u touches most outside region
vertex v touches most inside region

1. $k \leftarrow 0$
2. for each neighbor w of u touches the most outside region
3. TRAVERSAL_GRAPH(w, u, leftmost)
4. Update($G[k]$), $k \leftarrow k+1$
5. TRAVERSAL_GRAPH(w, u, rightmost)
6. Update($G[k]$), $k \leftarrow k+1$
7. for each neighbor w of v touches the most inside region
8. TRAVERSAL_GRAPH(w, v, leftmost)
9. Update($G[k]$), $k \leftarrow k+1$
10. TRAVERSAL_GRAPH(w, v, rightmost)
11. Update($G[k]$)
12. lexicographic sort $G[0]$, $G[1], \ldots, G[7]$

Figure 4.4: Algorithm for Normal Graph Representations

TRAVERSAL_GRAPH

Input: integer label of vertex f: from vertex f
integer label of vertex t to vertex t
direction: leftmost or rightmost

1. *from* $\leftarrow f$, *to* $\leftarrow t$
2. $n \leftarrow 0$
3. **for** $i = 0$ **to** 125
4. *to* $\leftarrow$ (*to*+ i) mod 126
5. **if** vertex labled *to* is WHITE
6. DFS(*from*, *to*, *direction*)
7. *from* $\leftarrow$ *to*

Figure 4.5: Algorithm for Traversal of a Graph

DFS

Input: label of vertex *from*
label of vertex u
direction: leftmost or rightmost

1. if color of vertex u is BLACK, **return**
2. color of $u \leftarrow$ BLACK
3. $F[u] = n$
4. $n \leftarrow n + 1$
5. **if** direction = leftmost version
6. **for** all unvisited vertex v adjacent to u in clockwise order
7. DFS(u, v, leftmost)
8. **else if** direction = rightmost
9. **for** all unvisited vertex v adjacent to u in counter-clockwise order
10. DFS(u, v, rightmost)

Figure 4.6: Algorithm of Leftmost and Rightmost DFS

We can compare the first normal graph representation of V with that of the SSM 7-Venn diagrams we have ready obtained, to see whether V is a new diagram.

Chapter 5

Conclusion

We have implemented two computer programs in order to compute all the simple symmetric monotone (SSM) Venn diagrams with seven curves. One program uses Grünbaum encoding to identify the diagrams (Algrithm One, Chapter 4) and the other program uses graph representations to identify the SSM diagrams (Algorithm Two, Chapter 4). In each program, there are 20,125,224 matrices generated. The total number of different SSM Venn diagrams we obtained is 23. The two programs come out with the exactly the same solutions. These 23 SSM diagrams match with the 23 SSM Venn diagrams found by A. Edwards, B. Grünbaum and F. Ruskey. These findings verified in practice that the Grünbaum encoding is indeed a representation of an SSM Venn diagram, which we have proved in theory in Chapter 2.

Index of Solutions	Index of Matrices
1	96,948
2	144,637
3	145,004
4	150,449
5	245,226
6	270,995
7	337,325
8	420,957
9	487,352
10	650,806
11	740,224
12	746,788
13	746,888
14	751,414
15	1,097,622
16	1,126,308
17	1,349,320
18	1,467,557
19	2,218,590
20	2,219,552
21	2,247,989
22	2,314,011
23	2,782,864

Bibliography

[1] Arthur T. White, Lowell W. Beineke. "Topological Graph Theory," *Selected Topics in Graph Theory*, ed. Lowell W. Beineke, Robin J. Wilson, Academic Press, 1978.

[2] B. Bultena, B. Grünbaum, F. Ruskey. "Convex Drawings of Intersecting Families of Simple Closed Curves," preprint.

[3] T. Cormen, C. Leiserson, R. Rivest. *Introduction to Algorithms*, McGraw-Hill Book Company, 1990.

[4] R. Diestel. *Graph Theory*, Graduate Texts in Mathematics 173, Springer-Verlag 1995.

[5] A. W. F. Edwards. "Seven-set Venn Diagrams with Rotational and Polar Symmetry," *Combinatorics, Probability, and Computing*, 7 (1998) 149-152.

[6] P. A. Firby, C. F. Gardiner. *Surface Topology*, second edition, Ellis Horwood Limited, 1991.

[7] B. Grünbaum. "Venn Diagrams and Independent Families of Sets," *Mathematics Magazine*, Jan-Feb 1975, 13-23.

[8] B. Grünbaum. "On Venn Diagrams and the Counting of Regions," *The College Mathematics Journal*, 15 (1984) 433-435.

[9] H. B. Griffiths. *Surfaces*, second edition, Cambridge University Press, 1981.

[10] P. Hamburger. "Doodles and Doilies," preprint.

[11] W. S. Massey. *Algebraic Topology: An Introduction*, Harcourt Brace Jovanovich, New York, 1967.

[12] F. Ruskey. "A Survey of Venn Diagrams," *Electronic Journal of Combinatorics*, 4 (1997) DS#5.

[13] J. Venn. "On the diagrammatic and mechanical representation of propositions and reasonings," *The London, Edinburgh, Dublin Philosophical Magazine and Journal of Science*, 9 (1880) 1 - 18.

VITA

Surname: Cao Given Names: Tao

Place of Birth: Wuhan, P. R. China

Educational Institutions Attended:

University of Victioria	1990 to 2000
Wuhan University	1982 to 1986

Degree Awarded:

M.Sc.	University of Victoria	1992
B.Sc.	Wuhan University	1986

Honours and Awards:

University of Victoria Fellowship	1990-1995
People's Prize	1982-1986

PARTIAL COPYRIGHT LICENSE

I hereby grant the right to lend my thesis to users of the University of Victoria Library, and to make single copies only for such users or in response to a request from the Library of any other university, or similar institution, on its behalf or for one of its users. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by me or a member of the University designated by me. it is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Title of Thesis:

Computing All the Simple Symmetric Monotone Venn Diagrams on Seven Curves

Autho

TAO CAO
(Name in Block Letter)

Jan 2, 2002
(Date)