Title Picture Theory: Algorithms and Software

Name Andrea Donafee

Picture Theory:

Algorithms and Software

Andrea Donafee

Doctor of Philosophy

2003

University of Luton

Abstract

This thesis is concerned with developing and implementing algorithms based upon the geometry of pictures. Spherical pictures have been used in many areas of combinatorial group theory, and particularly, they have shown to be a useful method when studying the second homotopy module, π_2, of a presentation ([3],[4],[7],[12],[41] and [64]). Computational programs that implement picture theoretical and design algorithms could advance the areas in which picture theory can be used, due to the much faster time taken to derive results than that of manual calculations.

A variety of algorithms are presented. A data structure has been devised to represent spherical pictures. A method is given that verifies that a given data structure represents a picture, or set of pictures, over a group presentation. This method includes a new planarity testing algorithm, which can be performed on any graph.

A computational algorithm has been implemented that determines if a given presentation defines a group extension. This work is based upon the algorithm of Baik *et al.* [1] which has been developed using the theory of pictures.

A 3-presentation for a group G is given by $< \mathcal{P}, \mathbf{s} >$, where $\mathcal{P}$ is a presentation for G and $\mathbf{s}$ is a set of generators for π_2. The set $\mathbf{s}$ can be described in a number of ways. An algorithm is given that produces a generating set of spherical pictures for π_2 when $\mathbf{s}$ is given in the form of identity sequences. Conversely, if $\mathbf{s}$ is given in terms of spherical pictures, then the corresponding identity sequences that describe π_2 can be determined.

The above algorithms are contained in the Spherical PIcture Editor (SPICE). SPICE is a software package that enables a user to manually draw pictures over group presentations and, for these pictures, call the algorithms described above. It also contains a library of generating pictures for the non abelian groups of order at most 30. Furthermore, a method has been implemented that automatically draws a spherical picture from a corresponding identity sequence. Again, this new graph drawing technique can be performed on any arbitrary graph.

For My Mum and Dad

Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my thanks and gratitude to my supervisors, Dr Cartsen Maple and Professor Steve Pride for all their help, encouragement and time that they have given me during this project. They have helped me a great deal and I have learnt a lot from them, both professionally and personally. I am very greatful to have had the oppertunity to have worked with them.

My family have always supported (emotionally and financially!) and belived in me. Without this I wouldn't have been able to finish this work. Thank you for your confidence in me and for always being there. (Thank you especially for putting up with me when I was writing up and I thought I would never finish!)

Last but definately not least I would like to thank Yvonne and Melony. You were all always there to listen to me moan and to share a glass of wine with (even if we were in separate countries!). Somehow you all managed to take my mind of things and make me laugh - I'm very lucky to have friends like you.

Declaration

I declare that this thesis is my own unaided work. It is being submitted for the degree of Doctor of Philosophy at the University of Luton. It has not been submitted before any degree or examination in any other University.

Andrea Donafee

25th day of March, 2003.

Notation

General:

$\mathbf{x}$	an alphabet
W	a word on $\mathbf{x}$
W^{-1}	inverse of W
$\sim_{\mathbf{x}}$	equivalence of words relative to $\mathbf{x}$
$[W]_{\mathbf{x}}$	the equivalence class of W, relative to $\mathbf{x}$
$F(\mathbf{x})$	free group on $\mathbf{x}$
$\mathbf{r}$	a set of words on $\mathbf{x}$
$\mathcal{P} =< \mathbf{x}; \mathbf{r} >$	group presentation
$\sim_{\mathcal{P}}$	equivalence of words relative to $\mathcal{P}$
$[W]_{\mathcal{P}}$	the equivalence class of W relative to $\mathcal{P}$
$G(\mathcal{P})$	group defined by $\mathbf{P}$
$N(\mathbf{r})$	the smallest normal subgroup of $\{[R] : R \in \mathbf{r}\}$
$\pi_2(\mathcal{P})$	second homotopy module of $\mathcal{P}$
s	the generating set of $\pi_2(\mathcal{P})$
$< \mathcal{P}; \mathbf{s} >$	a *3-presentation* for G
H	an extension of the groups K by Q
$\mathcal{H}$	a group presentation for H
$\mathcal{H}^{\circ}$	a "partial presentation" for H
$\hat{\mathcal{H}}$	a reduced presentation for H

Spherical Pictures:

Let $\mathbf{P}$ be a spherical picture over $\mathcal{P}$:

D^2	a closed ambient disc containing $\mathbf{P}$
$\delta\mathbf{P}$	boundary of D^2
$\mathcal{O}_\mathbf{P}$	a global basepoint of $\mathbf{P}$ on $\delta\mathbf{P}$
α	the collection of arcs in $\mathbf{P}$
δ	the collection of discs in $\mathbf{P}$
$\delta\Delta_i$	boundary of Δ_i ($\Delta_i \in \Delta$)
O_i	basepoint of Δ_i
$\epsilon(\Delta_i)$	the orientation of Δ_i ($\epsilon = \pm 1$)
$\epsilon(\Delta_i)R(\Delta_i)$	the label of Δ_i ($\epsilon = \pm 1$)
c	a corner of Δ_i
$W(c)$	the word obtained by a complete walk around Δ_i begining at c
β	transverse path in $\mathbf{P}$
$< \mathbf{P} >$	equivalence class containing the picture $\mathbf{P}$
$\mathbf{P_1} + \mathbf{P_2}$	the picture depicting the union of $\mathbf{P_1}$ and $\mathbf{P_2}$
$-\mathbf{P}$	the inverse picture of $\mathbf{P}$
γ	a spray on $\mathbf{P}$
$W(\gamma_i)$	the label of γ_i ($\gamma_i \in \gamma$)
$Pict_x(\mathcal{H})$	a collection of spherical pictures over the presentation $\mathcal{H}$
$\mathbf{t}(\mathcal{H})$	a collection of test words for admissibility of $\mathcal{H}$

Identity Sequences:

$\mathbf{w}$	the set of all words on $\mathbf{x} \cup \mathbf{x^{-1}}$
$\mathbf{r^w}$	words of the form $W\mathbf{r}^\epsilon W^{-1}$ ($W \in \mathbf{w}, \epsilon = \pm 1$)
θ	a sequence of words from $\mathbf{r^w}$
θ^{-1}	the inverse of θ
$\delta\theta$	the element of the free group $F(\mathbf{x})$ on $\mathbf{x}$
$< \theta >$	the equivalence class containing the sequence θ

Θ	the set of all equivalence classes for all such sequences θ

Complexes:

Γ	a 1-complex
$\mathbf{v}$	vertex set
$\mathbf{e}$	edge set
$\iota(e)$	initial vertex of edge e
$\tau(e)$	terminal vertex of edge e
e^{-1}	the inverse of edge e
$\{e, e^{-1}\}$	undirected edge
e^{+}	an orientation of $\mathbf{e}$
$deg(v)$	degree on v
Σ	a path in Γ of length N
Σ^{-1}	the inverse path of Σ
Σ_i	the i-th connection point in Σ $(1 \leq i \leq N)$
C	a cycle of Γ
C^{-1}	the inverse of C
$\mathbf{v}$'	a subset of $\mathbf{v}$
$\mathbf{e}$'	a subset of $\mathbf{e}$
$\Gamma' =< \mathbf{v}'; \mathbf{e}' >$	a subgraph of Γ
$\mathcal{S}$	a topological space that contains Γ'
$\phi(e)$	label of edge e
$\mathtt{real}(\Gamma)$	the realisation of Γ
ρ	a region of Γ
K^2	2-complex
$\mathcal{F}$	a collection of 2-cells of K^2
K^1	the 1-skelton of K^2
$\Pi(K^2)$	set of all paths in K^2
$\pi(K^2)$	fundamental groupoid of K^2

$\Pi(K^2, v)$	the set of all loops at vertex v
$\pi(K^2, v)$	fundamental group of K^2 at vertex v
$f_{\mathbf{P}}$	a map from sphere S^2 to K^2

Data Structures:

Π_Δ	the rotation scheme for Δ
Π_{Δ_i}	the local rotation scheme for disc Δ_i ($\Delta_i \in \Delta$)
$\sigma_{i,j}$	the connection point of Δ_i at arc numbered j
$W(\sigma_{i,j})$	the label of the connection point $\sigma_{i,j}$
$A(\Delta_i)$	the adjacency list that stores the rotation scheme Π_{Δ_i}
`RotationScheme`	the collection of adjacency lists Π_Δ
`RelatorInformation`	the list storing records of information for each $r \in \mathbf{r}$
$\mathcal{L}$	a list to store information
$\mathcal{L}_i$	the i-th component of $\mathcal{L}$
`Boundary`	a collection of lists used to store the position of arcs in the `PictureFromIdentitySequence` algorithm
CLeft	the left counter used in the `PictureFromIdentitySequence` algorithm
CRight	the right counter used in the `PictureFromIdentitySequence` algorithm
`RegionLabel`	the list containing the words associated with each region of $\mathbf{P}$
`AccessArcs`	the list containing the access arc to each region of $\mathbf{P}$
Υ_x	the correct order, around O, of all discs whose paths enter $\mathbf{P}$ through boundary region ρ_x
κ_i	the correct order, around O, of all discs in search path i
`IdentitySequence`	a list containing an identity sequence of $\mathbf{P}$
(x_c, y_c)	the coordinates at the centre of a picture disc
ϑ_i	an initial position of the endpoint of an arc ($1 \leq i \leq 8$)

r	the radius of a picture disc
(x_M, y_M)	the coordinates of the midpoint of a line, l, of an arc
M	a matrix used to construct the Picture Visibility representation of $\mathbf{P}$
ξ	a set of discs from Π_Δ

Chapter 1

Preliminaries

This thesis is concerned with developing and implementing algorithms based upon the geometry of pictures. Picture theory is an effective tool in the study of groups and has been used in many areas of combinatorial group theory. For instance, pictures have been studied with relation to the second homotopy module of a group presentation and investigating aspherical presentations (see [3],[4],[7],[12],[41] and [64]). Pictures have also been employed to express the cohomology of certain groups in [3] and [66], and in the study of Cockcroft presentations, [47]. Howie ([39],[40]), used pictures to determine a range of properties of one-relator product of groups, where the relator is a proper power.

The focus of this chapter is to provide background theory of group presentations and introduce some geometric techniques of combinatorial group theory. A detailed description of the work presented in this thesis is also outlined.

1.1 Words

Definition 1.1 (Word) *Let* $\mathbf{x}$ *be an alphabet, and* $\mathbf{x}^{-1}$ *be a set disjoint from* $\mathbf{x}$ *and in one-to-one correspondence,* $x \leftrightarrow x^{-1}$ $(x \in \mathbf{x})$, *with* $\mathbf{x}$. *A (group) word,* W, *on* $\mathbf{x}$ *is a sequence* $W = x_1^{\epsilon_1} x_2^{\epsilon_2} ... x_n^{\epsilon_n}$ *where* $n \geq 0$, $x_i \in \mathbf{x}$ *and* $\epsilon_i = \pm 1$ $(1 \leq i \leq n)$. *When* $n = 0$ *the word is the empty word, which is denoted by 1. The length of* W *is* n. *The inverse of* W *is the word*

$W^{-1} = x_n^{-\epsilon_n} ... x_1^{-\epsilon_1}$.

A *cyclic permutation* of W is a word of the form $x_p^{\epsilon_p} ... x_n^{\epsilon_n} x_1^{\epsilon_1} ... x_{p-1}^{\epsilon_{p-1}}$ for some p, $1 \leq p \leq n$. A *subword* of W is a (possibly empty) word of the form $x_r^{\epsilon_r} ... x_s^{\epsilon_s}$ for $1 \leq r \leq s \leq n$. Let $V = y_1^{\delta_1} y_2^{\delta_2} ... y_m^{\delta_m}$ ($y_i \in X$, $\delta_i = \pm 1$) be a word in $\mathbf{x}$. Then $W = V$ if and only if $n = m$, $x_i = y_i$ and $\epsilon_i = \delta_i$ for all $1 \leq i \leq n$. The product of WV is given by $x_1^{\epsilon_1} x_2^{\epsilon_2} ... x_n^{\epsilon_n} y_1^{\delta_1} y_2^{\delta_2} ... y_m^{\delta_m}$. Note that a word U is a *subword* of W if $W = W_1 U W_2$ for some words W_1 and W_2.

1.2 Free Groups

The following elementary operations can be performed on words:

(A_1) : If W contains a subword $x^\epsilon x^{-\epsilon}$ ($x \in \mathbf{x}, \epsilon = \pm 1$) then delete it.

$(A_1)^{-1}$: Insert a word $x^\epsilon x^{-\epsilon}$ at any position in W.

Two words, W_1, W_2, are said to be *freely equivalent*, written $W_1 \sim_{\mathbf{x}} W_2$, if W_2 can be obtained from W_1 by a finite series of elementary operations of the form $(A_1)^{\pm 1}$. The free equivalence class of a word W is denoted by $[W]_{\mathbf{x}}$ (or just $[W]$). The multiplication on the equivalence classes $[W_1][W_2] = [W_1 W_2]$ is well defined and, as such, the set of all equivalence classes combined with this multiplication give a group. The identity of this group is [1] and the inverse, $[W]^{-1}$, of $[W]$ is $[W^{-1}]$. This group is called the *free group* on $\mathbf{x}$ and is denoted by $F(\mathbf{x})$. A word is called *freely reduced* if no operation (A_1) can be applied to it. Each free equivalence class contains a *unique* freely reduced word [53], [55].

1.3 Group Presentations

A convenient way to specify a group is by means of a presentation. A group presentation is a pair, $\mathcal{P} = < \mathbf{x}; \mathbf{r} >$, where $\mathbf{x}$ is an alphabet and $\mathbf{r}$ is a set of words on $\mathbf{x}$. The elements of $\mathbf{x}$ are called the *generating symbols* and those of $\mathbf{r}$ are called the *defining relators*. Two new operations are introduced, in addition to $(A_1)^{\pm 1}$:

(A_2) : If W contains a subword R^ϵ $(R \in \mathbf{r}, \epsilon = \pm 1)$ then delete it.

$(A_2)^{-1}$: Insert R^ϵ $(R \in \mathbf{r}, \epsilon = \pm 1)$ at any position in W.

Two words W_1 and W_2 on $\mathbf{x}$ are said to be *equivalent* (relative to $\mathcal{P}$), denoted $W_1 \sim_{\mathcal{P}} W_2$, if W_2 can be obtained from W_1 by a finite series of elementary operations of the form $(A_1)^{\pm 1}$, $(A_2)^{\pm 1}$. Note that $\sim_{\mathcal{P}}$ is an equivalence relation on the set of all words on $\mathbf{x}$. The equivalence class containing W is denoted $[W]_{\mathcal{P}}$. The multiplication on the equivalence classes $[W_1]_{\mathcal{P}}[W_2]_{\mathcal{P}} = [W_1 W_2]_{\mathcal{P}}$ is well defined and as such, the set of all equivalence classes combined with this multiplication give a group. The identity of this group is $[1]_{\mathcal{P}}$ and the inverse, $[W]_{\mathcal{P}}^{-1}$, of $[W]_{\mathcal{P}}$ is $[W^{-1}]_{\mathcal{P}}$. This group is the *group defined by* $\mathcal{P}$ and is denoted $G(\mathcal{P})$.

There is a homomorphism from $F(\mathbf{x})$ onto $G(\mathcal{P})$ which is defined by

$$[W]_{\mathbf{x}} \to [W]_{\mathcal{P}} \qquad (W \text{ a word on } \mathbf{x}).$$

The kernel of this homomorphism is the smallest normal subgroup, $N(\mathbf{r})$, containing $\{[R] : R \in \mathbf{r}\}$. Therefore (by the first isomorphism theorem) there is an isomorphism

$$\frac{F(\mathbf{x})}{N(\mathbf{r})} \to G(\mathcal{P}) \qquad [W]_{\mathbf{x}} N(\mathbf{r}) \to [W]_{\mathcal{P}} \qquad (W \text{ a word on } \mathbf{x}).$$

Given a group G, $\mathcal{P} =< \mathbf{x}; \mathbf{r} >$ is a *presentation for* G if there is an isomorphism $G(\mathcal{P}) \to G$. If $\mathbf{x}$ is finite then G is *finitely generated*; if $\mathbf{r}$ is finite then G is *finitely related*; if both $\mathbf{x}$ and $\mathbf{r}$ are finite then G is *finitely presented*.

1.4 Examples of Presentations of Popular Groups

The following list provides examples of presentations of popular groups. For further examples, and details, see Appendix C, [44] and [55].

Cyclic groups: A presentation for the finite cyclic group of order n, C_n, is given by

$< y; y^n >$.

Symmetric groups: Let $S_n = \text{Sym}(\{1, 2, \cdots, n\})$ denote the symmetric group of degree n. If x_i corresponds to the transposition $(i, i+1)$ for $1 \leq i \leq n-1$, then a presentation for S_n is given by $< x_1, \cdots, x_{n-1}; \mathbf{r}, \mathbf{s}, \mathbf{t} >$ where

$\mathbf{r} = \{x_i^2 : 1 \leq i \leq n-1\}$,

$\mathbf{s} = \{(x_i x_{i+1})^3 : 1 \leq i \leq n-2\}$,

$\mathbf{t} = \{[x_i, x_j] : 1 \leq i < j-1 < n-1\}$.

When $n = 3$ the presentation becomes (using a for x_1 and b for x_2):

$$\mathcal{A} =< a, b, c; a^2, b^2, (ab)^3 > . \tag{1.1}$$

This presentation will often be referred to for illustrative purposes.

The Klein Four Group: A presentation for the Klein Four Group, $V = C_2 \times C_2$, is given by $< x, y; x^2, y^2, (xy)^2 >$.

Quaternion groups: The quaternion group, Q, consists of the eight quaternions $\pm 1, \pm i, \pm j, \pm k$. It is an non-abelian group of order 8 and has a presentation given by

$$< x, y; x^4, x^2 y^{-2}, y^{-1} x y x >,$$

where $x = \begin{pmatrix} i & 0 \\ 0 & -i \end{pmatrix}$ and $y = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$.

Dihedral groups: The dihedral group, D_n is the group of symmetries for an n-sided regular polygon. A presentation for the dihedral group D_n is $< x, y; x^2, y^n, (xy)^2 >$, where x corresponds to reflection and y corresponds to rotation by $\frac{2\pi}{n}$.

1.5 Pictures over Group Presentations

For a presentation, $\mathcal{P} = < \mathbf{x}; \mathbf{r} >$, a picture $\mathbf{P}$ over $\mathcal{P}$ is a geometric configuration with the following features:

1. A closed ambient disc D^2 with a basepoint $O_{\mathbf{P}}$ on the boundary of D^2. The boundary of D^2 is denoted by $\partial\mathbf{P}$. A clockwise orientation is selected for the ambient disc.

2. A collection of finite disjoint closed discs, $\Delta_1, \Delta_2, ..., \Delta_m$, contained in the interior of D^2. A basepoint, O_i, lies on the boundary, $\partial\Delta_i$, of Δ_i $(1 \leq i \leq m)$. Each disc is orientated clockwise.

3. A finite number of disjoint arcs, $\alpha_1, ..., \alpha_n$, which lie in the closure of $D^2 - \{\bigcup_{i=1}^{n} \Delta_i\}$. An arc can either be a circle or a line segment, and has a normal orientation indicated by a short arrow which meets the arc transversely. Each arc of $\mathbf{P}$ is labelled by an element of $\mathbf{x}$. Basepoints cannot lie on an arc of $\mathbf{P}$. Travelling around Δ_i from the basepoint in a clockwise direction, a succession of arcs are reached. The total label on these arcs must be a word, $R(\Delta_i)^{\epsilon_i(\Delta)}$, where $R(\Delta_i) \in \mathbf{r}$ and $\epsilon_i = \pm 1$. The disc Δ_i is then labelled by $\epsilon_i(\Delta_i)R(\Delta_i)$.

The *empty picture* is one which contains no arcs nor discs. If $\bigcup_i \Delta_i \cup \bigcup_j \alpha_j$ is connected then $\mathbf{P}$ is *connected*. The *components* of $\mathbf{P}$ are the connected parts of $\bigcup_i \Delta_i \cup \bigcup_j \alpha_j$. The *regions* of $\mathbf{P}$ are the closures of the connected components of $D^2 - (\bigcup_i \Delta_i \cup \bigcup_j \alpha_j)$. A region of $\mathbf{P}$ which is incident with $\partial\mathbf{P}$ is a *boundary region* of $\mathbf{P}$, while the remaining regions are called *interior regions*. If no arc of $\mathbf{P}$ meets $\partial\mathbf{P}$ then $\mathbf{P}$ is *spherical*, and in this case $\partial\mathbf{P}$ is usually not drawn. Throughout this work, unless otherwise stated, the term 'picture' will be used to refer to a spherical picture. Figures 1.1 and 1.2 illustrate a non-spherical and a spherical picture over $\mathcal{A}$, as defined in (1.1), where the relators a^2, b^2 and $(ab)^3$ are denoted respectively by R, S, T respectively. Note that the closed arc, labelled by a, in the top right-hand corner of the picture in Figure 1.1 is a *floating circle* and will be discussed on Page 8.

Reading the labels as they are encountered on a complete journey around $\partial\mathbf{P}$ in a clockwise direction from $O_{\mathbf{P}}$, gives the *boundary label* of $\mathbf{P}$. The boundary label of the picture in Figure 1.1 is $a^{-1}b^{-2}a^{-1} \sim_{\mathcal{P}} 1$. A *corner* of a disc Δ_i is a connected component of $\partial\Delta_i - \bigcup_j \alpha_j$. A word $W(c)$ is obtained by reading the labels on the arcs encountered on a walk around $\partial\Delta_i$ in a clockwise direction, beginning and ending at a corner c. The corner

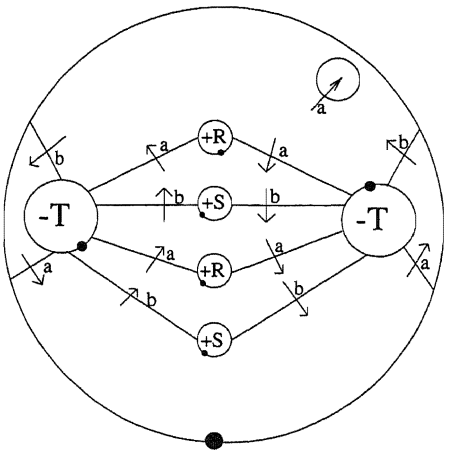


Figure 1.1: A non-spherical picture over $\mathcal{A}$

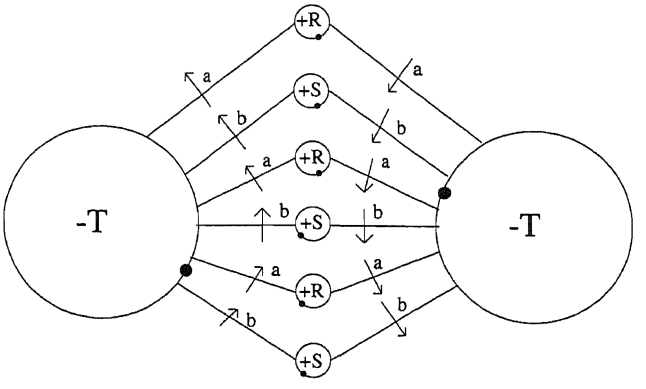


Figure 1.2: A spherical picture over $\mathcal{A}$

c is a *basic corner* if $W(c)$ is identically equal to $\epsilon(\Delta_i)R(\Delta_i)$.

Pictures are the duals of van Kampen diagrams and we can therefore apply the following pictorial version of the van Kampen lemma to pictures:

Lemma 1.5.1 *Let $\mathcal{P} =< \mathbf{x}; \mathbf{r} >$ be a group presentation, and let W be a word on $\mathbf{x}$. Then $W \sim_{\mathcal{P}} 1$ if and only if there is a picture over $\mathcal{P}$ with boundary label W.*

(A proof of this result is given in [4] and involves noting that each loop in $\mathbf{P}$ determines an element of the normal closure of $\mathbf{r}$ in $F(\mathbf{x})$.)

A *transverse path* of $\mathbf{P}$ is a simple path in the closure of $D^2 - \{\bigcup_{i=1}^{n} \Delta_i\}$ which crosses the arcs of $\mathbf{P}$ finitely many times. The dotted line in Figure 1.3 depicts a transverse path. If β is a closed transverse path in $\mathbf{P}$ then the interior area enclosed by β forms a *subpicture*, $\mathbf{B}$, of $\mathbf{P}$. An arbitrary basepoint, $O_{\mathbf{B}}$, can be selected on β, the boundary of $\mathbf{B}$, so that $\mathbf{B}$ is a

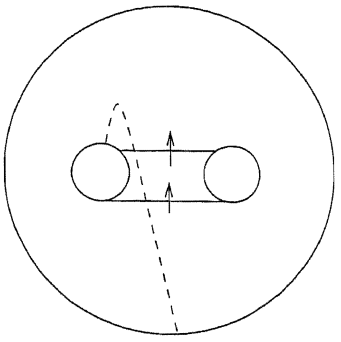


Figure 1.3: A transverse path

picture (see Figure 1.4 where the transverse path is again depicted by a dotted line).

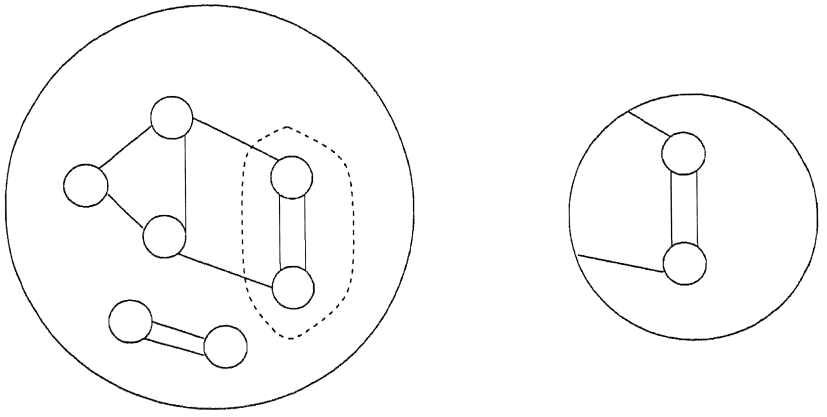


Figure 1.4: a) A picture **P** with a closed transverse path b) The subpicture, **B**, it encloses

The *complement* of **B** in **P** is a picture with the same boundary label as **B** and contains the discs of **P** - **B** with opposite signs; it is obtained by deleting **B** from **P** and performing a reflection on this picture, **P** - **B** (see [4], Figure V.3 for a drawing of the complement of a subpicture).

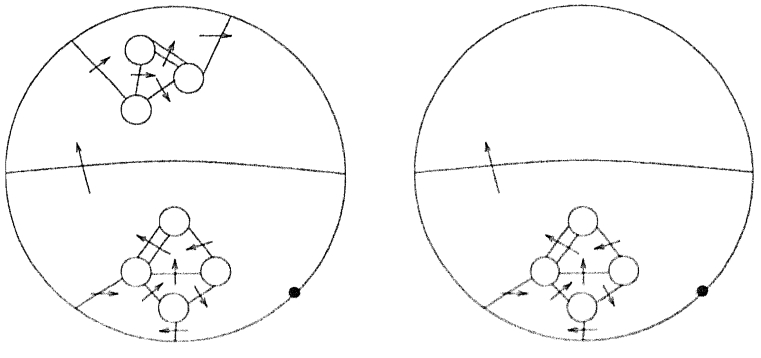


Figure 1.5: a) A free arc b) A floating arc

A *free arc* is an arc in **P** that does not intersect any disc, as shown in the non-spherical picture in Figure 1.5a. A *floating arc* is a free arc that does not enclose any non-trivial

picture, which is illustrated in Figure 1.5b. Note that a free closed arc is just a free circle. A *floating circle* is a free closed arc which encircles no discs or arcs of a picture, as is shown in the top right hand area of the non-spherical picture in Figure 1.1.

A *dipole* is a connected subpicture with two distinct discs, Δ_1 and Δ_2 say, such that:

- The two discs are labelled by the same relator but have opposite signs.
- The basepoints of Δ_1 and Δ_2 lie in the same region of the picture.

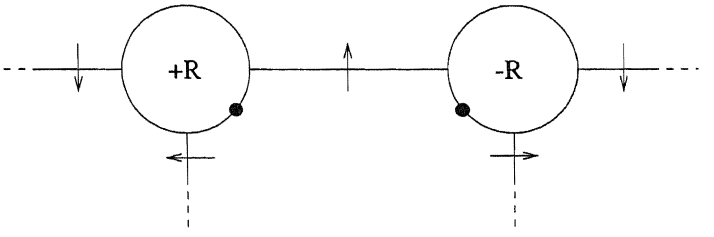


Figure 1.6: A dipole

Figure 1.6 depicts a dipole. A connected spherical picture over $\mathcal{P}$ that contains exactly two discs where each arc constitutes a dipole is referred to as a *complete dipole* over $\mathcal{P}$. Figure 1.7 illustrates a complete dipole.

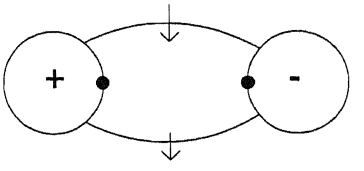


Figure 1.7: A complete dipole

1.6 The Second Homotopy Module and 3-Presentations

Operations can be performed on pictures. Two pictures over $\mathcal{P}$ are said to be *equivalent* if one can be obtained from the other by a finite number of the following elementary operations:

(1): Insert a *floating circle.*

$(1)^{-1}$: Delete a *floating circle.*

(2): Insert a complete dipole (see Figure 1.7).

$(2)^{-1}$: Delete a complete dipole.

(3): Perform a *bridge move.* A bridge move is illustrated in Figure 1.8.

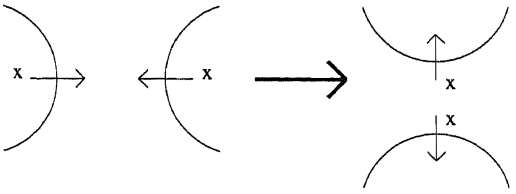


Figure 1.8: A bridge move

A *reduced* picture is a picture where no bridge moves can be performed on it to produce a subpicture which is a dipole. The above elementary operations lead to an equivalence relation on spherical pictures. The equivalence class containing the spherical picture $\mathbf{P}$ is denoted $< \mathbf{P} >$. The collection of equivalence classes has a module structure over the ring $\mathbf{Z}G$, where $G = G(\mathcal{P})$. This module structure is described as follows.

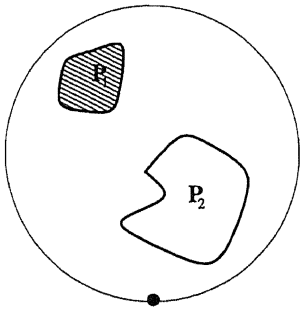


Figure 1.9: The picture $\mathbf{P_1} + \mathbf{P_2}$

Let $\mathbf{P_1}$ and $\mathbf{P_2}$ be spherical pictures over a group presentation $\mathcal{P}$. The picture $\mathbf{P_1} + \mathbf{P_2}$ can be produced, as is shown in Figure 1.9. Clearly, if the pictures $\mathbf{P_1'}$ and $\mathbf{P_2'}$ are equivalent to $\mathbf{P_1}$ and $\mathbf{P_2}$ respectively, then $\mathbf{P_1'} + \mathbf{P_2'}$ is equivalent to $\mathbf{P_1} + \mathbf{P_2}$. This therefore gives a well-defined addition: $< \mathbf{P_1} > + < \mathbf{P_2} > = < \mathbf{P_1} + \mathbf{P_2} >$. The collection of equivalence classes together with this addition form an abelian group. The zero of this group is the equivalence class of the empty picture and the negative, $-< \mathbf{P} >$, of $< \mathbf{P} >$ is the equivalence class $< -\mathbf{P} >$, where $-\mathbf{P}$ is the mirror image of $\mathbf{P}$. Figure 1.10 shows the negative picture of Figure 1.2.

It now remains to describe the action of $G = G(\mathcal{P})$. Let W be a word on $\mathbf{x}$, and let $\mathbf{P}$ be a spherical picture. The picture obtained from $\mathbf{P}$ by surrounding it by a sequence of

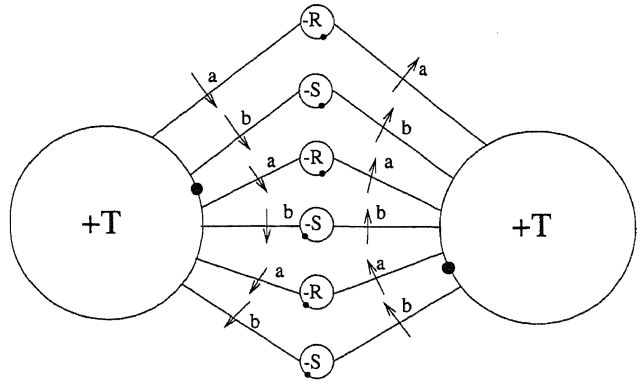


Figure 1.10: The negative picture of Figure 1.2

concatenating circles (as shown in Figure 1.11) with the label W is denoted $W \cdot \mathbf{P}$. The multiplication

$$[W]_{\mathcal{P}} \cdot < \mathbf{P} > = < W \cdot \mathbf{P} >$$

is well defined (see [64]). The set of equivalence classes of spherical pictures, with the above module structure, is called the *second homotopy module* of $\mathcal{P}$, which is denoted $\pi_2(\mathcal{P})$.

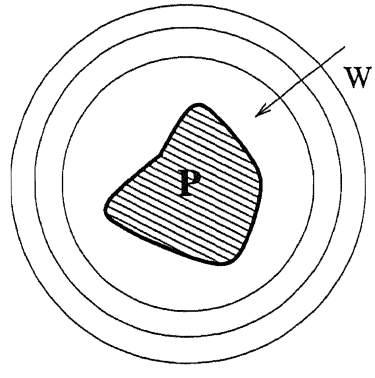


Figure 1.11: The picture $W \cdot \mathbf{P}$

Now let $\mathbf{s}$ be a set of spherical pictures over $\mathcal{P}$. The following additional operations are introduced:

(4): Insert a picture from $\mathbf{s} \cup -\mathbf{s}$.

$(4)^{-1}$: Delete a picture from $\mathbf{s} \cup -\mathbf{s}$.

Two pictures are said to be *equivalent modulo* $\mathbf{s}$, if one can be transformed to the other by a finite number of operations $(1)^{\pm 1}, (2)^{\pm 1}, (3), (4)^{\pm 1}$. A set $\mathbf{s}$ is said to be a *generating set*

of pictures if every spherical picture is equivalent modulo $\mathbf{s}$ to the empty picture. It can be shown, [64], that $\mathbf{s}$ is a set of generating pictures if and only if $\{< S >: S \in \mathbf{s}\}$ is a set of module generators of the module $\pi_2(\mathcal{P})$.

Definition 1.2 (3-Presentation) *A 3-presentation is a pair* $< \mathcal{P}; \mathbf{s} >$ *where* $\mathcal{P}$ *is a group presentation and* $\mathbf{s}$ *is a generating set of pictures for* $\pi_2(\mathcal{P})$.

1.7 Identity Sequences over Group Presentations

Let the set of all words on $\mathbf{x} \cup \mathbf{x^{-1}}$ be denoted by $\mathbf{w}$, and $\mathbf{r^w}$ denote the set of words of the form $WR^{\epsilon}W^{-1}$ where $W \in \mathbf{w}$, $R \in \mathbf{r}$ and $\epsilon = \pm 1$. *Sequences*, θ, of the form $(c_1, c_2, \cdots, c_n)$, where $c_i \in \mathbf{r^w}$ for all $1 \leq i \leq n$, are considered. The inverse, θ^{-1}, of θ is defined to be $(c_n^{-1}, \cdots, c_2^{-1}, c_1^{-1})$. Let $\delta\theta$ be the element $[c_1 c_2 \cdots c_n]$ of the free group $F(\mathbf{x})$ on $\mathbf{x}$. A sequence θ is said to be an *identity sequence* if and only if $\delta\theta = [1]$.

Certain elementary operations can be performed on sequences:

Exchange: Replace two consecutive terms c_i, c_{i+1} with either $c_{i+1}, c_{i+1}^{-1} c_i c_{i+1}$ or $c_i c_{i+1} c_i^{-1}, c_i$.

Deletion: Delete a consecutive pair of terms if one term is identically equal to the inverse of the other (that is, one term is of the form $WR^{\epsilon}W^{-1}$ and the other is $WR^{-\epsilon}W^{-1}$).

Insertion: Insert a consecutive pair of terms where one term is identically equal to the inverse of the other (i.e. insertion is the opposite of deletion).

Substitution: Replace each word, W, by a word freely equivalent to it.

Two sequences are said to be *Peiffer equivalent* if one can be obtained from the other by a finite number of these elementary operations. The equivalence class containing θ is denoted by $< \theta >$. Let $\circ$ be a binary operation on the set, Θ, of all equivalence classes, defined by

$$< \theta_1 > \circ < \theta_2 > = < \theta_1 \theta_2 > .$$

It can be shown, [64], that this operation is well defined. Under this binary operation

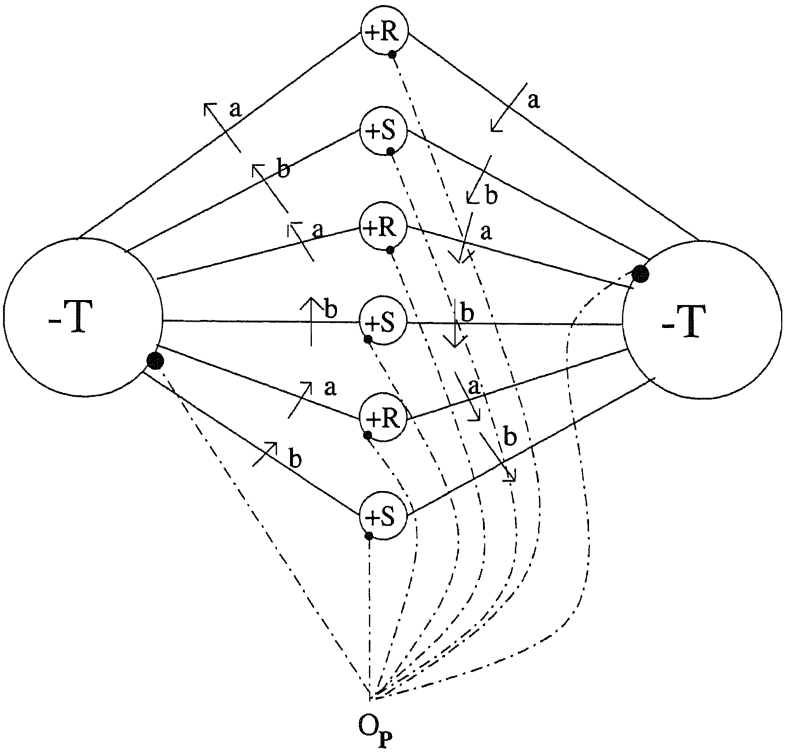


Figure 1.12: A spray for a picture of S_3

the elements of Θ form a group with the identity given by the equivalence class of empty sequences, and the inverse $< \theta >^{-1}$ of θ being $< \theta^{-1} >$. The subgroup of Θ which contains all equivalence classes of identity sequences is abelian. Moreover, there is a well-defined action of $G = G(\mathcal{P})$ on this subgroup given by

$$[W]_{\mathcal{P}} \cdot < \theta >=< W\theta W^{-1} >$$

where θ is an identity sequence $(c_1, c_2, \cdots, c_n)$ and

$$W\theta W^{-1} = (Wc_1W^{-1}, Wc_2W^{-1}, \cdots, Wc_nW^{-1}).$$

A $\mathbf{Z}G$ module is then obtained, called the *module of identity sequences* of $\mathcal{P}$.

Every identity sequence can be represented by a spherical picture, as is shown in [7], [27] and [64]. Conversely, an identity sequence can be obtained from a spherical picture $\mathbf{P}$, by creating a *spray* as follows. If $\mathbf{P}$ has n discs, $\Delta_1, ..., \Delta_n$, a spray is a sequence $\gamma = (\gamma_1, \gamma_2, ..., \gamma_n)$ of transverse paths such that each γ_i $(1 \leq i \leq n)$ starts at the basepoint, $O_{\mathbf{P}}$, and ends at the

basepoint of the corresponding disc, Δ_i. The word on γ_i, obtained by reading the labels of the arcs from the initial to the terminal point of γ_i, is denoted by $W(\gamma_i)$. Additionally, this sequence of paths only intersect at $O_{\mathbf{P}}$. Travelling around $O_{\mathbf{P}}$ in a clockwise direction in $\mathbf{P}$, the paths are encountered in order $\gamma_1, \gamma_2, ..., \gamma_n$. The identity sequence associated with $\mathbf{P}$ is then:

$$\theta(\mathbf{P}) = (W(\gamma_1)R(\Delta_1)^{\epsilon_1(\Delta_1)}W(\gamma_1)^{-1}, \cdots, W(\gamma_n)R(\Delta_n)^{\epsilon_n(\Delta_n)}W(\gamma_n)^{-1}) \tag{1.2}$$

Example 1.1

The dashed line in Figure 1.12 depicts a spray for the picture in Figure 1.12. This diagram gives the following identity sequence:

$$\begin{aligned}(\quad & bT^{-1}b^{-1}, S, b^{-1}Rb, b^{-1}a^{-1}Sab, b^{-1}a^{-1}b^{-1}Rbab, b^{-1}a^{-1}b^{-1}a^{-1}Sabab, \\ & b^{-1}a^{-1}b^{-1}a^{-1}b^{-1}Rbabab, b^{-1}a^{-1}b^{-1}a^{-1}T^{-1}baba).\end{aligned} \tag{1.3}$$

It can be shown, [27], that the mapping

$$< \mathbf{P} > \quad \mapsto \quad < \theta(\mathbf{P}) > \qquad\qquad (\mathbf{P} \text{ a spherical picture on } \mathcal{P})$$

is a well-defined module isomorphism from $\pi_2(\mathcal{P})$ to the module of identity sequences. Consequently identity sequences can be used to provide an alternative algebraic description of it 3-presentations:

Definition 1.3 (Algebraic 3-presentation) *A 3-presentation of a group G consists of a presentation $\mathcal{P} = < \mathbf{x}; \mathbf{r} >$ for G together with a set $\mathbf{s}$ of identity sequences such that the set*

$$\{< \sigma >: \sigma \in \mathbf{s}\}$$

generates the module of identity sequences.

1.8 Complexes

A *1-complex*, Γ, is a *graph* (in the sense of Serre [70]) which consists of a finite collection of vertices, $\mathbf{v}$, edges, $\mathbf{e}$, and the following three maps:

$$\begin{aligned} \iota &: \mathbf{e} \to \mathbf{v} \\ \tau &: \mathbf{e} \to \mathbf{v} \\ {}^{-1} &: \mathbf{e} \to \mathbf{e} \end{aligned}$$

The *initial* vertex of an edge, $e \in \mathbf{e}$, is denoted by $\iota(e)$ whilst $\tau(e)$ denotes the *terminal* vertex of e. The initial and terminal vertices of e are referred to as the *extremities* of e. The *inverse edge* of e is e^{-1}, where $\iota(e^{-1}) = \tau(e)$, $\tau(e^{-1}) = \iota(e)$, $e^{-1} \neq e$ and $(e^{-1})^{-1} = e$. The pair of inverse edges $\{e, e^{-1}\}$ depicts an *undirected edge*. An *orientation* $\mathbf{e}^+$ of $\mathbf{e}$ consists of a choice of one element from each edge pair $\{e, e^{-1}\}$. Given that an orientation has been selected for all edges incident on a vertex, v, the number of edges incident on a v is the *degree* of the vertex, and is denoted $deg(v)$.

A *path*, Σ, in Γ is a collection of edges such that $\Sigma = (e_1, e_2, \cdots, e_n)$ and $\iota(e_{i+1}) = \tau(e_i)$, for $1 \leq i < n$. If any two vertices are the extremities of some path, Γ is said to be *connected*. The length of Σ is n, with the vertices $\iota(\Sigma) = \iota(e_1)$ and $\tau(\Sigma) = \tau(e_n)$ forming the extremities of Σ. The path is a *cycle*, (or *loop*, or *closed path*) if $\iota(\Sigma) = \tau(\Sigma)$. The *inverse path* of Σ, denoted Σ^{-1}, is the path $(e_n^{-1}, e_{n-1}^{-1}, \cdots, e_1^{-1})$. A *simple path* is one where $\iota(e_i) \neq \iota(e_j)$ and $\tau(e_i) \neq \tau(e_j)$ for all $i \neq j$. A *simple cycle* is a simple path that begins and ends at the same vertex. An edge, e, is a *self loop* if $\iota(e) = \tau(e)$. A path is *reduced* if $e_i \neq e_{i+1}^{-1}$ $(1 \leq i < n)$. A cycle is *cyclically reduced* if it is reduced and $e_1 \neq e_n^{-1}$. A *simple graph* is a graph with no self loops and only one edge pair between any two vertices.

A *subgraph* of Γ consists of subsets $\mathbf{v}', \mathbf{e}'$ of $\mathbf{v}$ and $\mathbf{e}$ respectively, such that for any $e' \in \mathbf{e}'$, $\iota(e'), \tau(e') \in \mathbf{v}'$ and $e'^{-1} \in \mathbf{e}'$.

A *labelled graph* is a graph with an extra function $\phi : \mathbf{e} \to G$, where G is a group and $\phi(e^{-1}) = \phi(e)^{-1}$, for all $e \in \mathbf{e}$.

A *realisation* (also referred to as a *graph drawing*), $\texttt{real}(\Gamma)$ of Γ, is a function such that each vertex, $v \in \mathbf{v}$, is mapped to a distinct point $\texttt{real}(\Gamma(v))$ in a topological space and each edge, $e \in \mathbf{e}$, is a simple open Jordan curve $\texttt{real}(\Gamma(e))$ with endpoints $\texttt{real}(\Gamma(\iota(e)))$ and $\texttt{real}(\Gamma(\tau(e)))$. For two distinct edges $e, f \in \mathbf{e}$, the curves $\texttt{real}(\Gamma(e))$ and $\texttt{real}(\Gamma(f))$ intersect at most at their endpoints. Generally, the realisation of Γ involves electing an orientation and Γ is then said to be an *oriented graph.* The graph drawing of Γ involves only the edges in $\mathbf{e}^+$, with an arrow indicating the direction from $\iota(e)$ to $\tau(e)$, $(e \in \mathbf{e}^+)$. Travelling along an edge e against the direction of the arrow depicts the edge e^{-1}. An oriented edge, e, is shown in Figure 1.13.

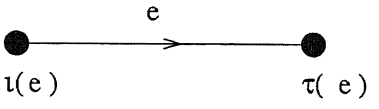


Figure 1.13: An oriented edge

A *2-complex*, K^2, consists of a 1-complex, K^1, a set of 2-cells (faces), $\mathcal{F}$, and two functions, δ and $^{-1}$, defined on $\mathcal{F}$. The function δ maps the boundary of a face, $\mathcal{F}_i$ say, to a cycle in K^1, whilst $^{-1}$ assigns an inverse face, $\mathcal{F}_i^{-1}$, to $\mathcal{F}_i$. It is required that $\delta(\mathcal{F}_i^{-1}) = \delta(\mathcal{F}_i)^{-1}$. See [36], Figure I.3 for a drawing of a 2-complex.

Group presentations can be used to define 2-complexes (for example, see [36], Section 1.3). Given a presentation $\mathcal{P} =< \mathbf{x}; \mathbf{r} >$, the corresponding 2-complex has a single 0-cell (that is, a vertex), an edge pair x, x^{-1} for each $x \in \mathbf{x}$ and a pair of 2-cells, $\mathcal{F}_R, \mathcal{F}_R^{-1}$ for each $R \in \mathbf{r}$, with $\delta(\mathcal{F}_R^{\pm 1}) = R^{\pm 1}$. Figure 1.14 illustrates the edge pairs (on the left-hand side of the diagram) and 2-cells of a 2-complex for the presentation, $\mathcal{A}$, given in (1.1).

The 1-complex, K^1, is referred to as the *1-skeleton* of K^2 and a path in K^2 is a path in the 1-skeleton of K^2. The set of all paths in K^2 is denoted by $\Pi(K^2)$. Two paths in K^2 are said to be equivalent if one can be obtained from the other by a finite number of insertions or deletions of either the parts of the form $\{e, e^{-1}\}$ or boundary paths of a face $\mathcal{F}_i^{\pm 1}$. The quotient of $\Pi(K^2)$ by this equivalence relation forms the *fundamental groupoid*, $\pi(K^2)$, of K^2. The semigroup $\Pi(K^2, v)$ consists of all loops at vertex v. The *fundamental group*, $\pi(K^2, v)$ of K^2 at the point v is the image of $\Pi(K^2, v)$ in $\pi(K^2)$.

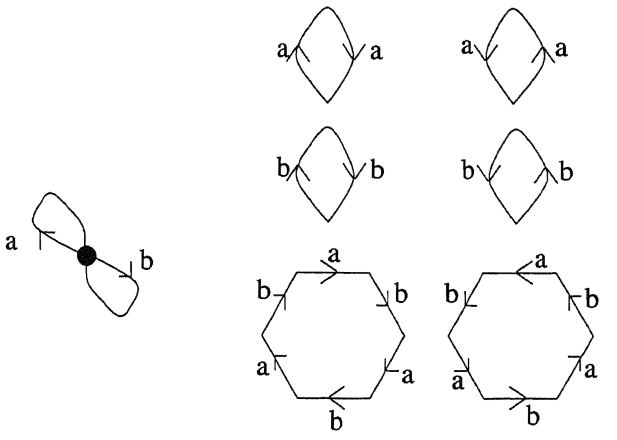


Figure 1.14: A drawing of a 2-complex for $\mathcal{A}$

Bogley and Pride [4] show how to obtain a map $f_{\mathbf{P}} : (D^2, \delta D^2) \to (K^2, K^1)$ for each picture $\mathbf{P}$ over $\mathcal{P}$. If two pictures are equivalent (relative to X) then they lie in the same *homotopy class*. The homotopy class of $\mathbf{P}$ is given by $[f_{\mathbf{P}}]$ and this class is an element of the relative homotopy group $\pi_2(K^2, K^1)$. When $\mathbf{P}$ is spherical, there is the pair $(S^2, f_{\mathbf{P}})$ where $f_{\mathbf{P}}$ is a map from the sphere S^2 to K^2 that preserves dimension and incidence and $[f_{\mathbf{P}}]$ is an element of $\pi_2(\mathcal{P}) = \pi_2(K^2)$. Pictures can therefore be used to represent elements of the relative homotopy group $\pi_2(K^2, K^1)$ and spherical pictures can depict elements of the absolute homotopy group $\pi_2(K^2)$.

1.9 The GAP System

The two most widely used computational discrete algebra systems are Groups, Algorithms and Programming (GAP) [28] and Magma [54]. The algorithms presented in this work have been implemented in GAP, using version 4.2. The main reason for using GAP is that it is a free system, unlike Magma, and as such the algorithms presented in this work are readily available to all.

1.9.1 Features of GAP

GAP consists of five parts:

- The **kernel**, written in the programming language C, which implements the GAP programming language. The kernel also provides an interactive way to develop and use GAP programs.
- An accessible **library of functions** that contains the algebraic algorithms.
- A collection of **share packages** which have been developed by independent groups. Share packages can be written solely in the GAP language or may include external C programs.
- A **data library** containing various group libraries including those of small groups, character tables, ATLAS of finite groups and a library of tables of marks.
- The **documentation** which is available on-line or as a down-loadable file and has reference, programming, tutorial and extending GAP manuals.

GAP is mainly concerned with computational group theory, although it is by no means restricted to this. It contains procedures for investigating the properties of groups, vector spaces, modules and algebras. For a more detailed look at the capabilities of GAP see the GAP 4 Manual [29]. An extensive bibliography of all the literature that utilises the GAP system is given in [28].

1.10 Scope and Rationale

One of the main uses of pictures is to study the structure of the second homotopy module, $\pi_2(\mathcal{P})$, of a group presentation $\mathcal{P}$. The manner in which pictures represent elements of $\pi_2(\mathcal{P})$ will be described in some detail in Chapter 3. There are many references for the basic theory, an overview is given in the articles [7], [12] and [64]. Bogley and Pride [4] present a survey article which documents ways in which to obtain generators of the second homotopy group of a presentation for various group constructions, such as HNN extensions, direct products, free products with amalgamation and split extensions. Indeed, by expressing $\pi_2(\mathcal{P})$ generators in terms of pictures, Baik and others [1] have added to this calculus by providing a way to calculate second homotopy module generators of general group extensions.

Representing elements of $\pi_2(\mathcal{P})$ geometrically has also been used to determine if a presentation is *aspherical*. An *aspherical* presentation occurs when $\pi_2(\mathcal{P})$ is trivial. A presentation is *combinatorially aspherical*, (CA), if $\pi_2(\mathcal{P})$ is generated by based spherical pictures over $\mathcal{P}$ that contain exactly two discs. A background to CA presentations is presented in [4]. Hitchman [35] used pictures to characterise an N-identity property for 2 complexes. From this he developed a method to test if a pair of 2-complexes (Y, X) were CA when the complex Y is obtained from X by the addition of a 2-cell. The conditions required for reduced presentations of group extensions to be aspherical has been investigated in [1].

For relative group presentations, authors use the term aspherical instead of CA. Bogley and Pride [3] give the criteria for asphericity of the relative presentation, $< G, t; atbtct >$ for $G =< a, b, c >$. Indeed this article also uses pictures to determine the homology and cohomology of the group G for dimensions greater than 2. Similarly, Howie and Metaftsis [41] investigate the relative group presentation of the form $< G, t; atbtctdtet >$ where a, b, c, d, e are generators of G, and find conditions under which it is aspherical. *Quasi-asphericity* occurs when there are no reduced spherical pictures containing at least one disc over $\mathcal{P}$. Edjvet and Thomas [23] use the fact that if a presentation is quasi-aspherical then $\pi_2(\mathcal{P})$ is generated by dipoles; from this they prove that the groups $(l, m|n, k)$, defined by presentations of the form $< a, b; a^l, b^m, (ab)^n, (ab^{-1})^k >$, are infinite for 'large' values of l, m, n, k.

Other uses of picture theory can be seen in the work of Howie [39], [40], who used geometric techniques to determine many properties of one-relator product of groups for the relator $R = S^m$ where S is a cyclically reduced word of length at least two and $m \geq 4$. Additionally his work with Duncan [22], using pictures to prove a conjecture of Weinbaum, improves these results to $m \geq 3$. Furthermore, pictures have been used to study the relational modules of groups with presentations that have exactly two types of generator in each relator [66].

As the calculus for the manipulation of pictures is so well developed for these group constructions, in these situations pictures could be seen as a more useful tool to work with than van Kampen diagrams. Indeed, in several applications using these geometric methods

is extremely effective; for example, computing identities among relations (see [12],[64]) and determining if a given presentation defines a group extension [1]. Problems may arise, however, when constructing these pictures. For instance, consider the case where a presentation has many relators of large length. A picture over this presentation would contain many discs, each with a high degree. To construct the correct picture over this presentation is an extremely time consuming task; this problem may limit the uses of pictures in combinatorial group theory. If a fast and efficient method was found to produce pictures over presentations, pictures themselves could be studied more. This would, in turn, enable pictures to be produced over different group constructions, and perhaps extend the applications of picture theory.

Clearly, a software package using picture theoretical techniques to compute properties of group presentations and provides an editor for the drawing of spherical pictures would be of use. In this thesis such work is presented. Chapter 2 describes the computer data structure used to represent spherical pictures. This chapter is then concerned with verifying that a representation does depict a picture over a given presentation. This 'verification' stage includes a new planarity test that can be used on an arbitrary graph.

Despite the fact that they can be extremely useful, no computational algorithms have been developed that use picture theoretical techniques. A method that produces a set of pictures from an algebraic description of π_2, and vice versa, is given in Chapter 3. Chapter 4 presents an algorithm which determines if a given presentation defines a group extension. This algorithm has been developed from the geometric techniques presented in [1].

The remainder of the thesis looks at developing the software editor for spherical pictures. Recall that pictures are essentially planar, possibly non-simple graphs. Currently, graphs can be studied in GAP using the share package GRAPE [72]. GRAPE has been utilised in several areas - in applications to combinatorial topology [67] and design theory [73]. GRAPE is intended for the construction and analysis of finite graphs related to groups. It cannot, however, visualise these graphs. Indeed many functions of GRAPE only work for simple graphs [74]. A graphical user interface for GAP is given with the XGAP share package

[9], but this also does not have the current capability to visualise pictures. In an email on the GAP forum Joachim Neubueser said that, "to write a program using XGAP that will display directed graphs would need the extension of the basic facilities of XGAP, namely the introduction of 'directed edges' as new basic graphical objects. This is possible, but it would mean a larger piece of work....." [62]. To produce drawings of pictures on screen, it was obviously necessary to develop this basic graphical object. Chapter 5 introduces the concept of directed arcs (cf. directed edges) and presents a technique that automatically draws a picture from a given representation. The capabilities and features of the editor are provided in Chapter 6. Chapter 7 draws some conclusions from the work presented in this thesis and looks at areas for further study.

Chapter 2

Planarity Testing

This chapter describes both the method in which pictures are represented and a technique to ensure that a given representation does give rise to a picture over a group presentation. A picture, $\mathbf{P}$, over a group presentation $\mathcal{P} =< \mathbf{x}; \mathbf{r} >$, can be thought of as a planar, possibly non-simple, graph comprising of a set of arcs, α, and a set of discs, Δ. The discs in these graphs have the additional features of a label, a basepoint and an orientation associated with them. The ordering of arcs emanating from each disc is clearly an integral feature of $\mathbf{P}$, and as such a *rotation scheme* can therefore be used to efficiently represent $\mathbf{P}$.

Definition 2.1 (Rotation Scheme) *A rotation scheme,* $\Pi = \{\Pi_{\Delta_i}; \Delta_i \in \Delta\}$, *is such that for each disc* Δ_i *of* $\mathbf{P}$ *there is a cyclic permutation,* Π_{Δ_i}, *of arcs incident on* Δ_i *which represents their circular (clockwise) order around* Δ_i.

It is necessary to devise a method to test that a given rotation scheme does indeed represent a picture over a group presentation. To reach this conclusion, two conditions must be checked. The first requirement is that the word obtained by reading the labels encountered when traversing each disc in the rotation scheme, Π, needs to be a cyclic permutation of a relator of $\mathcal{P}$. This can be achieved by a straight forward procedure. Let $W = x_1 \cdots x_n$ ($x_i \in \mathbf{x}, 1 \leq i \leq n$) be the word obtained by multiplying the labels of consecutive arcs incident on a disc, Δ_j, of Π say. If W does not equal a relator of $\mathcal{P}$ it is cyclically permuted

to obtain $W = x_2 \cdots x_n x_1$; if this does not equal a relator it is similarly permuted again. This process is repeated until either W equals a relator or again becomes equal to $x_1 \cdots x_n$. If the latter situation occurs then the word associated with Δ_j is not a relator of $\mathcal{P}$, and so Π cannot depict a picture over $\mathcal{P}$. This procedure is repeated for all discs in Π. This method, implemented in the function `CheckDiscs`, also provides a way by which to calculate possible locations of the basepoint of each disc, as shown in Chapter 3.

Determining the second requirement, that Π represents a planar graph, is more complex. This chapter is primarily concerned with describing such a planarity test. In this work the planarity testing algorithm is described with respect to pictures. It should be noted, however, that the algorithm can be used on general graphs and has wider ranging applications than picture theory.

2.1 Rotation Schemes

A rotation scheme is said to be *achievable* if it represents a planar graph. A *connection point*, $\sigma_{i,m}$, is the arc endpoint incident on the disc indexed by i connected at position m, relative to the basepoint. The generator that an arc depicts is given by traversing the arc clockwise around each disc it is incident upon. Due to this convention, the 'label' of an arc is given by a generator and its inverse. For instance $\{(\sigma_{i,m}, x), (\sigma_{j,n}, x^{-1})\}$, denotes the arc shown in Figure 2.1. The *local* rotation scheme for a disc Δ_i gives the disc that each arc on Δ_i is attached to, together with the generator that each connection point of Δ_i represents.

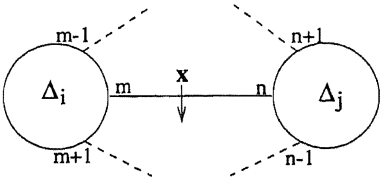


Figure 2.1: The arc $\{(\sigma_{i,m}, x), (\sigma_{j,n}, x^{-1})\}$

Example 2.1

Consider the group D_4, which has a presentation $< x, y; x^2, y^4, (ab)^2 >$. The relators can be denoted by R, S and T respectively. A picture over this presentation is given in Figure 2.2,

where the connection points of the picture have been labelled. Note that, for explanatory purposes, each disc and corner in Figure 2.2 is indexed. This picture has the following

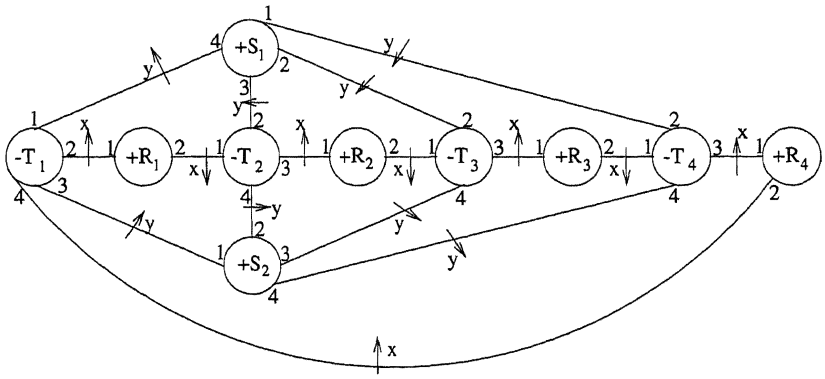


Figure 2.2: A picture over $< x, y; x^2, y^4, (ab)^2 >$

rotation scheme:

$$\Pi(+R_1) := \{(\sigma_{-T_1,2}, x), (\sigma_{-T_2,1}, x)\}$$

$$\Pi(+R_2) := \{(\sigma_{-T_2,3}, x), (\sigma_{-T_3,1}, x)\}$$

$$\Pi(+R_3) := \{(\sigma_{-T_3,3}, x), (\sigma_{-T_4,1}, x)\}$$

$$\Pi(+R_4) := \{(\sigma_{-T_4,3}, x), (\sigma_{-T_1,4}, x)\}$$

$$\Pi(+S_1) := \{(\sigma_{-T_4,2}, y), (\sigma_{-T_3,2}, y), (\sigma_{-T_2,2}, y), (\sigma_{-T_1,1}, y)\}$$

$$\Pi(+S_2) := \{(\sigma_{-T_1,3}, y), (\sigma_{-T_2,4}, y), (\sigma_{-T_3,4}, y), (\sigma_{-T_4,4}, y)\}$$

$$\Pi(-T_1) := \{(\sigma_{+S_1,4}, y^{-1}), (\sigma_{+R_1,1}, x^{-1}), (\sigma_{+S_2,1}, y^{-1}), (\sigma_{+R_4,2}, x^{-1})\}$$

$$\Pi(-T_2) := \{(\sigma_{+R_1,2}, x^{-1}), (\sigma_{+S_1,3}, y^{-1}), (\sigma_{+R_2,1}, x^{-1}), (\sigma_{+S_2,2}, y^{-1})\}$$

$$\Pi(-T_3) := \{(\sigma_{+R_2,2}, x^{-1}), (\sigma_{+S_1,2}, y^{-1}), (\sigma_{+R_3,1}, x^{-1}), (\sigma_{+S_2,3}, y^{-1})\}$$

$$\Pi(-T_4) := \{(\sigma_{+R_3,2}, x^{-1}), (\sigma_{+S_1,1}, y^{-1}), (\sigma_{+R_4,1}, x^{-1}), (\sigma_{+S_2,4}, y^{-1})\}$$

2.1.1 A Data Structure for Pictures

A common data structure used to represent graphs is an *adjacency matrix* [16].

Definition 2.2 (Adjacency Matrix) *Given an arbitrary graph with a collection of n*

vertices, an adjacency matrix, M, consists of an $n \times n$ array. The rows and columns of this matrix correspond to the vertices, with $M_{uv} = 1$ if there is an edge between vertices u and v and $M_{uv} = 0$ otherwise.

Matrices can therefore be constructed with both a column and a row representing every disc in a picture. The rotation scheme for each disc would then be given by the corresponding row. Assume that $\{(\sigma_{i,m}, x), (\sigma_{j,n}, x^{-1})\}$ was an arc in an arbitrary picture. The adjacency matrix that stores the rotation scheme for this picture would then have the value $[m, x]$ at row i column j and $[n, x^{-1}]$ at row j column i.

Example 2.1 (ctd.)

The adjacency matrix for the picture in Figure 2.2 is therefore:

	$+R_1$	$+R_2$	$+R_3$	$+R_4$	$+S_1$	$+S_1$	$-T_1$	$-T_2$	$-T_3$	$-T_4$
$+R_1$	0	0	0	0	0	0	[1,x]	[2,x]	0	0
$+R_2$	0	0	0	0	0	0	0	[1,x]	[2,x]	0
$+R_3$	0	0	0	0	0	0	0	0	[1,x]	[2,x]
$+R_4$	0	0	0	0	0	0	[2,x]	0	0	[1,x]
$+S_1$	0	0	0	0	0	0	[4,y]	[3,y]	[2,y]	[1,y]
$+S_1$	0	0	0	0	0	0	[1,y]	[2,y]	[3,y]	[4,y]
$-T_1$	$[2,x^{-1}]$	0	0	$[4,x^{-1}]$	$[1,y^{-1}]$	$[3,y^{-1}]$	0	0	0	0
$-T_2$	$[1,x^{-1}]$	$[3,x^{-1}]$	0	0	$[2,y^{-1}]$	$[4,y^{-1}]$	0	0	0	0
$-T_3$	0	$[1,x^{-1}]$	$[3,x^{-1}]$	0	$[2,y^{-1}]$	$[4,y^{-1}]$	0	0	0	0
$-T_4$	0	0	$[1,x^{-1}]$	$[3,x^{-1}]$	$[2,y^{-1}]$	$[4,y^{-1}]$	0	0	0	0

Problems arise, however, when more than one arc exists between the same two discs. This could be overcome by use of a multidimensional $|\Delta| \times |\Delta| \times k$ matrix, where Δ denotes the collection of discs in the picture and k is the maximum number of arcs between any two discs. This method, however, leads to complicated manipulation techniques. Indeed as the algorithms associated with this work normally involve examining every component in the matrix this leads to a computation time of $O(|\Delta^2| * k)$.

An *adjacency list* can also be used to represent graphs. The general definition is as follows:

Definition 2.3 (Adjacency List) *Given an arbitrary graph with a set* $\mathbf{v}$ *vertices, an adjacency list representation,* A*, consists of a collection of* $|\mathbf{v}|$ *lists. Each list,* A_u*, contains the list of edges incident on vertex* u*, for all* $u \in \mathbf{v}$*.*

Adjacency lists can be used to represent graphs with a rotation scheme (for example see [58], Chapter 8). Hence, adjacency lists can be used to store the rotation scheme of a picture. For every disc, Δ_i, in a picture $\mathbf{P}$, the adjacency list, $A(\Delta_i)$, for Δ_i stores the rotation scheme Π_{Δ_i}. Given the arc $\{(\sigma_{i,m}, x), (\sigma_{j,n}, x^{-1})\}$, the value in the mth position of $A(\Delta_i)$ is $[\sigma_{j,n}, x]$, which is the other connection point of the arc, together with the generator that $\sigma_{i,m}$ represents.

Example 2.1 (ctd.)

The adjacency list for the picture in Figure 2.2 is therefore:

$$
\begin{aligned}
A(+R_1) &:= [[-T_{1,2}, x], [-T_{2,1}, x]] \\
A(+R_2) &:= [[-T_{2,3}, x], [-T_{3,1}, x]] \\
A(+R_3) &:= [[-T_{3,3}, x], [-T_{4,1}, x]] \\
A(+R_4) &:= [[-T_{4,3}, x], [-T_{1,4}, x]] \\
A(+S_1) &:= [[-T_{4,2}, y], [-T_{3,2}, y], [-T_{2,2}, y], [-T_{1,1}, y]] \\
A(+S_2) &:= [[-T_{1,3}, y], [-T_{2,4}, y], [-T_{3,4}, y], [-T_{4,4}, y]] \\
A(-T_1) &:= [[+S_{1,4}, y^{-1}], [+R_{1,1}, x^{-1}], [+S_{2,1}, y^{-1}], [+R_{4,2}, x^{-1}]] \\
A(-T_2) &:= [[+R_{1,2}, x^{-1}], [+S_{1,3}, y^{-1}], [+R_{2,1}, x^{-1}], [+S_{2,2}, y^{-1}]] \\
A(-T_3) &:= [[+R_{2,2}, x^{-1}], [+S_{1,2}, y^{-1}], [+R_{3,1}, x^{-1}], [+S_{2,3}, y^{-1}]] \\
A(-T_4) &:= [[+R_{3,2}, x^{-1}], [+S_{1,1}, y^{-1}], [+R_{4,1}, x^{-1}], [+S_{2,4}, y^{-1}]]
\end{aligned}
$$

Adjacency lists can be produced without difficulty and enable data to be easily accessed. They are therefore an efficient data structure to store rotation schemes of pictures.

For a complete representation of a picture it is also necessary to store information about the orientation and labels of the discs. Given that each disc in the picture $\mathbf{P}$ over $\mathcal{P}$ can be

uniquely referred to, a record for each relator in $\mathcal{P}$ can be constructed with the following components:

- `Relator` which contains a relator of $\mathcal{P}$.
- `Label` which stores the label of `Relator`.
- `PositiveDiscs` which is a list of all discs in $\mathbf{P}$ labelled by `Label` and traversed in a clockwise direction.
- `NegativeDiscs` which is a list of all discs in $\mathbf{P}$ labelled by `Label` and traversed in an anti-clockwise direction.

These records can be stored in a list which will be referred to as the `RelatorInformation` list $\mathbf{P}$.

Example 2.1 (ctd.)

The `RelatorInformation` list for Figure 2.2 is:

$$\begin{aligned}[\; [\; & \texttt{Relator} := x^2, \texttt{PositiveDiscs} := [R_1, R_2, R_3, R_4], \texttt{NegativeDiscs} := [0], \texttt{Label} := R], \\ [\; & \texttt{Relator} := y^4, \texttt{PositiveDiscs} := [S_1, S_2], \texttt{NegativeDiscs} := [0], \texttt{Label} := S], \\ [\; & \texttt{Relator} := (xy)^2, \texttt{PositiveDiscs} := [0], \texttt{NegativeDiscs} := [T_1, T_2, T_3, T_4], \texttt{Label} := T]]\end{aligned}$$

2.2 Planarity Testing for Graphs Represented by a Rotation Scheme

2.2.1 Existing Planarity Testing Techniques

Planar graphs are extremely useful both theoretically and in real life applications. The theory of planar graphs has been examined extensively; one field in which it has proved particularly valuable is simplifying complex topological systems [16]. Moreover, only planar graphs are acceptable to represent certain systems. This is apparent in areas such as

communication networks [79],[42], circuit design [59], very large-scale integration layout (VLSI) [6],[52], and genetics [78], to name but a few.

Kuratowski [50] published the first results in planarity testing. He proved that a graph is planar if and only if it contains no subgraph isomorphic to either the complete graph on five vertices or a complete bipartite graph on six vertices. These graphs are shown in Figure 2.3. This result is extremely impractical to implement however, as testing for such subgraphs requires a computational time proportional to $|\mathbf{v}|^6$, where $|\mathbf{v}|$ is the number of vertices in the graph.

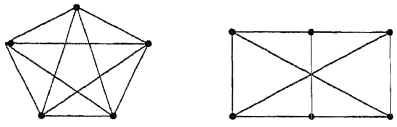


Figure 2.3: Kuratowski subgraphs

Hopcroft and Tarjan [37] developed the first planarity testing algorithm which had linear (in terms of vertices) time and space bounds. The authors employed a depth first search method to decompose the graph into subgraphs which were individually embedded into a surface. If all subgraphs of the graph could be embedded in this surface then the graph was said to be planar. An alternative to this 'path addition' approach, is the vertex addition method developed by Lempel *et al.* [51], which was later improved upon by Booth and Lueker [5] to give a linear computation bound. Recently, dynamic planarity testing algorithms have come under consideration due to the demands of important real life applications such as network optimisation, distributed computing and VLSI [20], where it is not unusual to require that graphs of existing systems be modified in some way. A technique has been developed by Di Battista and Tamassia [20] that determines in $O(\log |\mathbf{v}|)$ time if edges, or vertices and edges, can be inserted into a planar graph, Γ with $|\mathbf{v}|$ vertices, whilst maintaining planarity.

Once planarity has been determined, certain algorithms then proceed by computing a possible embedding for this graph. Chiba *et al.* [10] implement the embedding phase of the Booth and Lueker algorithm [5]. The Hopcroft and Tarjan algorithm [37] is expanded by Mehlhorn and Mutzel [57] to construct a planar embedding in linear time. In the latter technique, trees are constructed to represent subgraphs of the original graph. The properties of these

trees are manipulated and an embedding of the graph computed. A rotation scheme is used to represent this embedding. A rotation scheme is also employed by Mehlhorn [57] in the algorithm given to embed a graph in *any* arbitrary surface; this algorithm also has a linear time bound. Another method of interest is that of Shih and Hsu [71] whereby the examination of certain tree structures enables the recognition of planarity and the embedding of the graph to be computed simultaneously. Their method, as with all the algorithms referred to above, is valid only for *simple graphs*, graphs with only one edge between two vertices. The main reason for this is that the graphical data is represented, and manipulated, by a *tree*, a connected graph without a closed circuit; non-simple graphs can not therefore be depicted by a tree structure. Moreover, trees do not provide a way to store, and maintain, the order of edge incidence on vertices of a graph.

The planarity test presented here determines if the rotation scheme of a, possibly non-simple, graph, Γ, is achievable. If Γ is found to be planar, the regions of Γ are produced. Determining the regions of the picture therefore occurs concurrently with establishing planarity. Extracting regions from a graph has been examined previously by Fan and Chang [25] and Jiang and Bunke [43] with time complexities of $O(|\mathbf{v}|^2)$ and $O(|\mathbf{e}| \log |\mathbf{e}|)$ respectively (where $|\mathbf{v}|$ is the number of vertices and $|\mathbf{e}|$ the number of edges of the original graph). Both these algorithms take the coordinates of the vertices of a planar, simple, graph as input. A problem with this approach is that it is unusual for a graph to be represented by coordinates and thus these values would have to be calculated before this technique could be used. The main advantage of the algorithm proposed herein is that it is valid for any graph regardless of loops, multiple edges or connectivity and it is independent of a coordinate system.

2.2.2 Overview of the Planarity Testing Algorithm

The concept of the algorithm is to embed distinct paths of Γ into a topological space $\mathcal{S}$. It is assumed that Γ is constructed, as with pictures, with discs and arcs. This does not detract from the fact that Γ can be a general graph; discs can be thought of as vertices and arcs as edges. A path, Σ, of length N, is constructed with connection points where Σ_i

denotes the i-th connection point of Σ. The following protocol is used: if (Σ_{i-1}, Σ_i) defines a corner of a disc then (Σ_i, Σ_{i+1}) must characterise an arc of the graph. Similarly if (Σ_{i-1}, Σ_i) defines an arc then (Σ_i, Σ_{i+1}) must define a corner of a disc. Corners are constructed by two consecutive connection points on a disc, and by traversing the disc in a clockwise direction.

The first step is to remove from Γ any disc with only one arc emanating from it[1]. Clearly, discs with this property do not affect the planarity of Γ. Removing such discs may cause additional discs to now have degree one. These discs are iteratively extracted until every disc has degree greater than one or Γ is empty. If the latter occurs then Γ must be a tree, and as such is planar. If Γ is not a tree, then Γ must contain a simple cycle, C. This cycle C represents two regions of a topological space, $\mathcal{S}$. (Note that these regions are not necessarily the regions of the embedded graph Γ.) The regions defined by C are embedded into $\mathcal{S}$, and $\mathcal{S}$ now contains a graph Γ' say, defined by C.

A path, Σ, of Γ is computed that can be embedded in Γ'. It is required that the connection points of Σ are not contained in Γ'. In order for Σ to be embedded in Γ', the first and last connection points of Σ, must be associated with a disc contained in Γ'. The region of Γ' in which Σ_1 can be embedded is determined. It is assumed that (Σ_1, Σ_2) defines an arc rather than a corner of a disc, although the argument is essentially the same if (Σ_1, Σ_2) defines a corner. If Σ_2 is associated with a disc that is not contained in Γ', it is not yet embedded and the corner (Σ_2, Σ_3) and the arc (Σ_3, Σ_4) are calculated. This process is repeated until a connection point, Σ_N say, is associated with a disc in Γ'. If Σ_1 and Σ_N can be embedded in the same region of Γ' then Σ is embedded in Γ' and the new regions of Γ' are determined. If Σ_1 and Σ_N lie in different regions of Γ', however, then Σ can not be embedded without violating the planarity of Γ' and Γ has been found to be non-planar.

Whilst Γ' is planar, a path consisting of connection points of Γ not in Γ' is computed. The first and last connection points of this path must be associated with discs that have been embedded previously in Γ'. It is determined whether Σ can be embedded in Γ'. The procedure terminates when either Γ is found to be non-planar or every connection point of

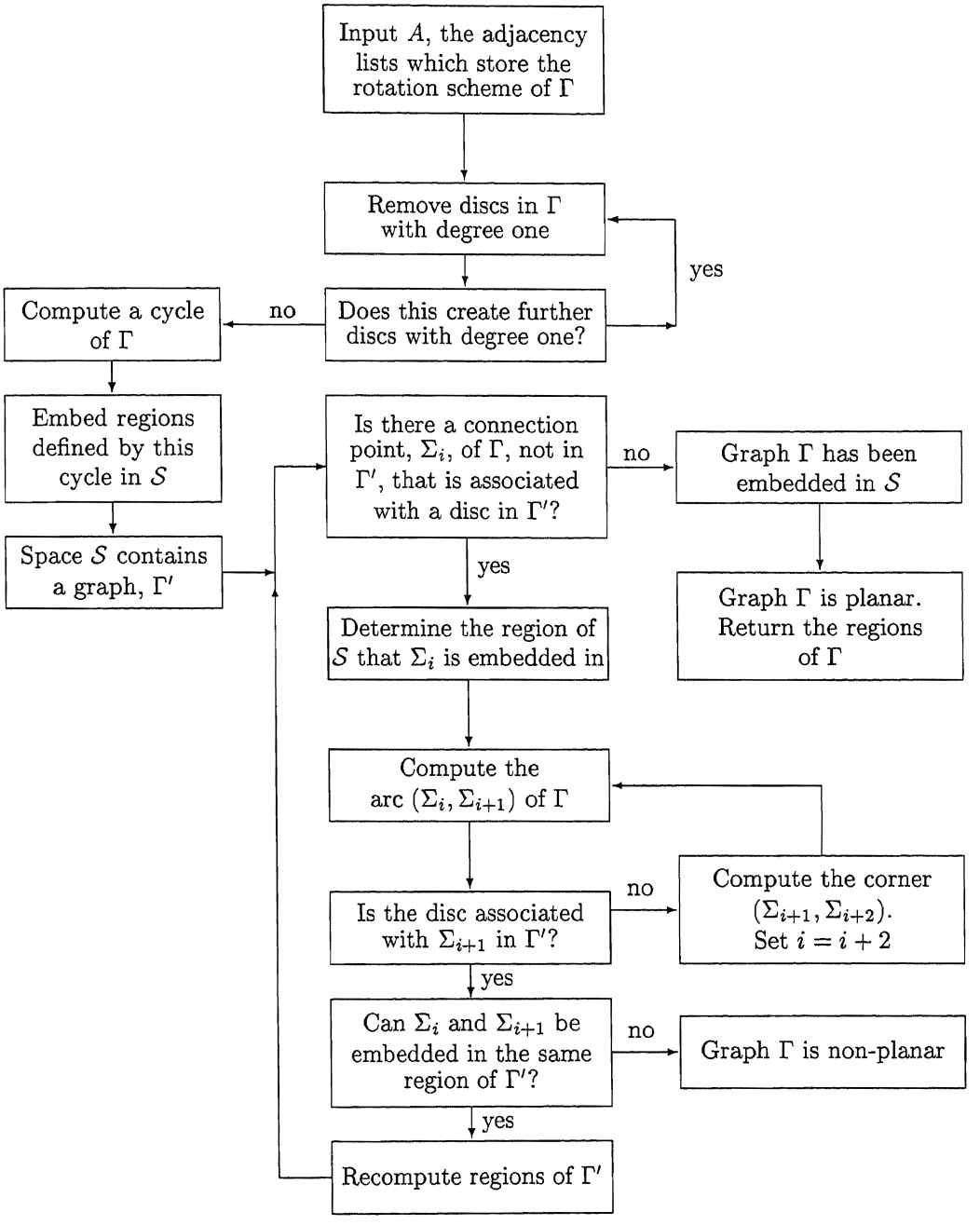
[1]Note that although this situation occurs in general graphs, it does not occur in pictures.

Γ has been embedded in Γ'. If all connection points of Γ can be embedded in $\mathcal{S}$, that is $\Gamma = \Gamma'$, then it is concluded that the graph Γ defines a planar graph and the regions of Γ are produced. The planarity testing algorithm is represented schematically in Figure 2.4.

This algorithm can be considered in three parts:

- Determining the regions that embedding a cycle of Γ into $\mathcal{S}$ creates.
- Computing and embedding paths of Γ into the graph, Γ'.
- Recomputing the regions of Γ' once a new path of Γ has been embedded into it.

Figure 2.4: The algorithm to determine planarity of Γ



2.2.3 Determining Regions from a Cycle

The representation of Γ is examined and all discs of degree one are (iteratively) removed due to the following proposition.

Proposition 2.2.1 *Discs with only one arc incident upon it do not affect the planarity of* Γ.

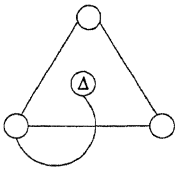


Figure 2.5: An edge crossing caused by a disc having degree one

The validity of this proposition can be illustrated by the graph drawing given in Figure 2.5. Here, the arc emanating from the disc, Δ, with degree one causes a crossing. Figure 2.6 demonstrates how the position of Δ can be altered so that no crossing occurs and the rotation scheme for Γ remains unchanged.

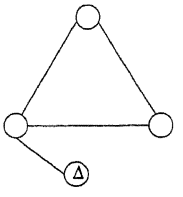


Figure 2.6: A planar graph with a disc having degree one

It is useful to summarise that properties of cycles. The results given in Proposition 2.2.2 are used to define the new regions created when a path has been embedded in a graph.

Proposition 2.2.2 *Let C be a simple cycle constructed using the above method and data structure. This cycle has the following properties:*

1. *Each disc in C occurs twice.*
2. *If a connection point, $\sigma_{\Delta,x}$, occurs in C then either the previous or the subsequent connection point traversed in a walk around C must be $\sigma_{\Delta,y}$ where $y = x \pm 1$, (modulo the degree of Δ).*

3. No two connection points traversed in a walk around C are the same.

Proof This proposition is proved by recalling that a cycle is described by both arcs and corners, with the convention that a corner must be traversed after an arc and, similarly, an arc must be traversed after a corner has been encountered. Recall that a corner of a disc, Δ say, is a section of the boundary of Δ that contains no other arcs emanating from Δ. Obviously a corner is therefore depicted by two connection points incident on the same disc, thus each disc in a cycle must occur at least twice.

Assume that a cycle contains more that two connection points incident on the same disc. This can only occur if either the path taken by the cycle leaves the disc and then revisits it, or the cycle travels across consecutive corners of the disc. These situations can not occur: C is a simple cycle that by definition contains distinct discs and the latter possibility contradicts how cycles are constructed. Each disc in a cycle therefore occurs exactly twice.

The two connection points of a disc, Δ, in a cycle must define the corner of $\delta\Delta$ in that cycle. The numbering of these connection points must therefore be consecutive (modulo the degree of Δ) due to the numbering of arcs by means of a rotation scheme.

If the two connection points of a disc in C are equal, this implies that a disc has only one corner, which can only occur if a disc has degree one. This situation is not allowed and thus the final part of the proposition is proved. □

A simple cycle, C, in Γ is determined and the regions defined by C are then embedded into $\mathcal{S}$. The space $\mathcal{S}$ now contains the graph, Γ', say. The regions of Γ' are determined as follows.

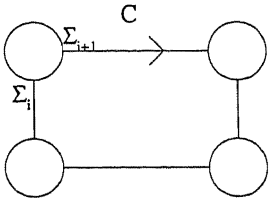


Figure 2.7: A diagram where C defines the exterior region

Lemma 2.2.1 *Let C be a simple cycle that has been embedded in space $\mathcal{S}$ to create a graph Γ'. The graph Γ' has two regions. These regions are defined by the connection points of C and C^{-1}.*

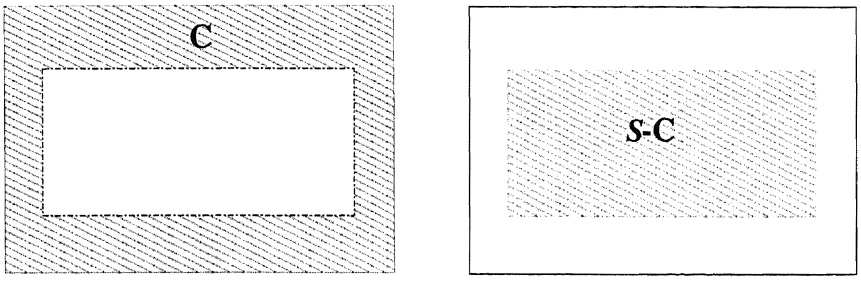


Figure 2.8: The two regions obtained by cutting along C

Proof Let C be a simple cycle that has been embedded in $\mathcal{S}$ to create a graph Γ', as shown in Figure 2.7. If the discs of C are contracted to a single point, Figure 2.8 shows how the path that C takes can be cut along to obtain two distinct parts: C and $\mathcal{S} - C$ (as indicated by the shaded parts). The connection points of C therefore define one region of Γ'. The space $\mathcal{S} - C$ is characterised by C^{-1}: if (Σ_i, Σ_{i+1}) is a corner of a disc in the region characterised by C, then (Σ_{i+1}, Σ_i) defines the section of the boundary of that disc not in that region. This is depicted in Figure 2.9. Embedding cycle C in $\mathcal{S}$ thus creates two regions of Γ', as required. □

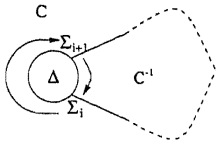


Figure 2.9: Regions bounded by the corners of Δ

2.2.4 Computing and Embedding Paths of Γ

Once C has been embedded into $\mathcal{S}$, the planarity of Γ is determined by embedding distinct paths of Γ into Γ'. There must be a disc, Δ_i say, that belongs to both Γ and Γ' with a connection point, Σ_1 say, incident upon it that is in Γ and not Γ'. If there is no such connection point then $\Gamma = \Gamma'$ and therefore Γ is a planar graph. The region of Γ' in which Σ_1 can be embedded into can be determined from the following result.

Lemma 2.2.2 *Let* $\rho = (\sigma_{\Delta_i,s}, \cdots, \sigma_{\Delta_w,x}, \sigma_{\Delta_w,z}, \cdots, \sigma_{\Delta_i,s+1 \bmod deg(\Delta_i)})$ *characterise a region of* Γ' *and let* $\Sigma_j = \sigma_{\Delta_w,y}$ *be a connection point of* Γ *that is not in* Γ'. *The connection point,* Σ_j, *can be embedded into* ρ *if and only if one of the following cases holds:*

Case 1 *If $x < z$ then Σ_j can only be embedded if $x < y < z$.*

Case 2 *If $x > z$ then Σ_j can only be embedded if either*

1. *$x > y$ and $y < z$ or*
2. *$x < y$ and $y > z$*

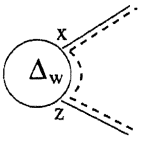


Figure 2.10: Embedding Σ_j in Γ'

Proof The proof of this lemma follows from the fact that every disc has a clockwise numeric ordering of arcs and that a connection point can only be embedded in an existing region if this ordering is preserved. The arcs around Δ_w must therefore be given in the clockwise order $\{x, y, z\}$, or a permutation of this ordering depending on the values of x and z. Figure 2.10 can be used to illustrate this idea. Here the dashed lines depict the only region that $\sigma_{w,y}$ can be embedded into whilst maintaining the order given by the rotation scheme of Δ_w. □

Having calculated the region in which to embed Σ_1, the corresponding arc $\Sigma = (\Sigma_1, \Sigma_2)$ can be calculated. If Σ_2 has a disc associated with it that is not in Γ', then the corner (Σ_2, Σ_3) and the arc (Σ_3, Σ_4) is computed. This process is repeated until a connection point, Σ_N say, is encountered that is associated with a disc in Γ'. The following result is established.

Theorem 2.2.1 *Let $\Sigma = (\Sigma_1, \cdots, \Sigma_N)$ be a path in Γ. If Σ_1 and Σ_N lie in different regions of Γ' then Γ is non-planar. Alternatively, if Σ_1 and Σ_N lie in the same region of Γ' then Σ can be embedded in Γ'.*

Proof Let Σ_1 and Σ_N be connection points that have been embedded, according to the conditions given in Lemma 2.2.2, in regions ρ and ρ' of Γ' respectively, where $\rho \neq \rho'$. As regions are bounded by both arcs and corners, obviously trying to embed Σ in Γ' involves crossing the arc(s) that bound ρ and ρ', which violates planarity of Γ'. As Γ' is a subgraph

of Γ this embedding would also violate planarity of Γ. If Σ_1 and Σ_N lie in the same region then Σ can be embedded into this region of Γ' whilst maintaining planarity. □

2.2.5 Recomputing Regions Upon the Embedding of a Path

Once a path of a graph Γ, Σ say, has been embedded in Γ' this both creates new regions of Γ' and changes existing ones. Let $\Sigma = (\sigma_{\Delta_x,b}, \cdots, \sigma_{\Delta_y,g})$ be the path that has be embedded in region ρ_i of Γ'. Let $\rho_i = (\sigma_{\Delta_x,a}, \sigma_{\Delta_x,c}, \xi_1, \sigma_{\Delta_y,f}, \sigma_{\Delta_y,h}, \xi_2)$ where ξ_j, (for $j = \{1,2\}$), denotes a (possibly empty) ordered set of connection points of ρ_i. The regions that the new embedding creates are defined by examining ρ_i, Σ and Σ^{-1}, the connection points of Σ in reverse order. Note that once Σ has been embedded, ρ_i is no longer a region of Γ', according to the definition of regions used in this work. The number of new regions created by the embedding is determined by the number of cycles in Σ. A possible drawing of this embedding is given in Figure 2.11.

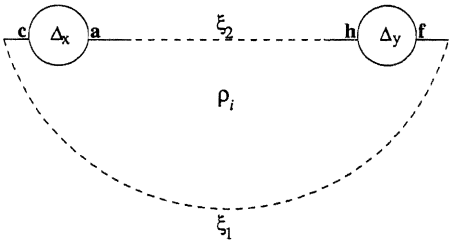
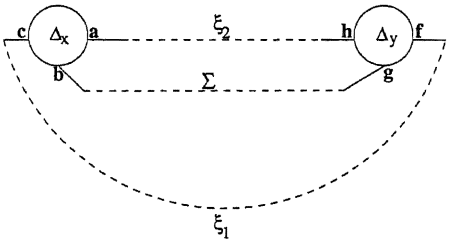


Figure 2.11: a) The region ρ_i



b) Embedding Σ in ρ_i

Lemma 2.2.3 *Let Σ be a path with m cycles that is to be embedded into Γ'. Embedding Σ into Γ' creates $m+2$ new regions of Γ'.*

Proof Let Σ be a path with that has been embedded into region ρ_i of Γ'. Assume that Σ has no cycles. The discs of ρ_i and Σ can be contracted to a single point. As Σ is simple this reduces it to one single line. This is depicted in Figure 2.12a. The path taken by Σ and ρ_i can be cut along to split ρ_i into two parts, as shown by the shaded regions of Figure 2.12b. As Σ is simple, $m = 0$ and therefore $m + 2$ new regions have been created by embedding Σ into ρ_i.

Figure 2.12: a) Contracting discs of ρ_i and Σ b) Cutting along Σ and ρ_i

Alternatively, let Σ be a non-simple path with m cycles. By applying the same argument as above, one can contract the discs of Σ to a single point and cut along the connection points of Σ. This divides ρ_i into $m+2$ separate regions. This is shown in Figure 2.13, where $m = 2$. The shaded regions in Figure 2.13b are the regions created by embedding Σ. Embedding a path into in a region of Γ' therefore creates $m + 2$ new regions, as required. □

Figure 2.13: a) Contracting discs of ρ_i and Σ b) Cutting along Σ and ρ_i

In order to define the connection points of the new regions created by embedding a path, the following results are required. Firstly, it is assumed that Σ is a simple path and that this path has been defined in ρ_i, where Σ and ρ_i are defined above.

Proposition 2.2.3 *If Σ is a simple path the regions defined by embedding this path in region ρ_i are as follows:*

$$\rho_{i,1} = (\sigma_{\Delta_x,a}, \Sigma, \sigma_{\Delta_y,h}, \xi_2)$$

$$\rho_{i,2} = (\sigma_{\Delta_x,c}, \xi_1, \sigma_{\Delta_y,f}, \Sigma^{-1})$$

Proof Embedding $\Sigma_1 = \sigma_{\Delta_x,b}$ into ρ_i creates a new corner of Δ_x, $\{\sigma_{\Delta_x,a}, \sigma_{\Delta_x,b}\}$. Similarly, the last connection point of Σ creates a new corner of Δ_y, $\{\sigma_{\Delta_y,g}, \sigma_{\Delta_y,h}\}$. The set ξ_2 proceeds $\sigma_{\Delta_y,h}$ and this set is itself preceded by $\sigma_{\Delta_x,a}$. This gives the set of connection points: $\{\sigma_{\Delta_x,a}, \Sigma, \sigma_{\Delta_y,h}, \xi_2\}$. Due to the way in which ρ_i and Σ were constructed, the connection points of this set fulfil the requirements of Proposition 2.2.2, and this set is therefore a

simple cycle of Γ'. From Lemma 2.2.1, it is known that the connection points of a simple cycle define one region of a graph. This cycle therefore defines a region of Γ'.

By applying a similar argument as above, embedding Σ creates two more new corners, $\{\sigma_{\Delta_x,b}, \sigma_{\Delta_x,c}\}$ and $\{\sigma_{\Delta_y,f}, \sigma_{\Delta_y,g}\}$. The set ξ_1 occurs after $\sigma_{\Delta_x,c}$ and before $\sigma_{\Delta_y,f}$ and the following path is formed: $\{\sigma_{\Delta_x,c}, \xi_1, \sigma_{\Delta_y,f}, \Sigma^{-1}\}$. Again, using the results of Proposition 2.2.2 and Lemma 2.2.1, it can be concluded that this path is a cycle that defines a region of Γ'.

Hence, the regions of Γ' created by embedding Σ in ρ_i are:

$$\begin{aligned} \rho_{i,1} &= (\sigma_{\Delta_x,a}, \Sigma, \sigma_{\Delta_y,h}, \xi_2) \\ \rho_{i,2} &= (\sigma_{\Delta_x,c}, \xi_1, \sigma_{\Delta_y,f}, \Sigma^{-1}), \end{aligned}$$

as required. □

If Σ is a non-simple path, the multiple regions that are created by embedding Σ are determined using the following result.

Proposition 2.2.4 *Let $\Sigma = (\Sigma_1, \Sigma_2, \cdots, \Sigma_n)$ be a non-simple path embedded in Γ'. Every disc in Σ is associated with at least two connection points, $\sigma_{\Delta,i}$ and $\sigma_{\Delta,j}$ say. Without loss of generality it is assumed that $i \leq j$. If $j \neq i+1 \mod deg(\Delta)$ for any disc associated with Σ^{-1}, then there exists a region, ρ_k, defined by the subcycle $(\Sigma_x, \cdots, \Sigma_y)$ of Γ' where $i < x < y < j$.*

If m is the number of cycles in Σ then this condition holds for m discs. The distinct connection points of Σ and Σ^{-1} not in these m cycles, together with the connection points of ρ_i, characterise two additional regions of Γ'.

Proof Let Σ be a path with m cycles. From Lemma 2.2.3 it is known that embedding Σ must create $m+2$ new regions of Γ'.

Assume that $m = 1$ and Δ occurs 4 times in Σ, where $\sigma_{\Delta,x}$ and $\sigma_{\Delta,w}$ are non-adjacent occurrences Δ. Given that $\sigma_{\Delta,w}$ occurs before $\sigma_{\Delta,x}$, Figure 2.14 depicts a possible drawing

of Δ.

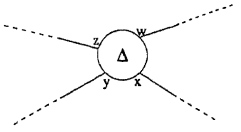


Figure 2.14: A possible drawing of a subsection of Σ

Let γ be a subsection of Σ^{-1} such that $\gamma = (\sigma_{\Delta,x}, \cdots, \sigma_{\Delta,w})$. Traversing γ and the corner $(\sigma_{\Delta,w}, \sigma_{\Delta,x})$ defines a simple cycle. This cycle must be simple as $\sigma_{\Delta,w}$ and $\sigma_{\Delta,x}$ are the only connection points in Σ that are not consecutive (modulo the length of Σ). This simple cycle, defined by γ, therefore defines a region of Γ'.

It is known from Proposition 2.2.3 that the connection points of γ also occur in region $\rho_{i,1}$. Due to the fact that a connection point can only appear in two regions then connection points of γ cannot occur in any other region.

Recall that the connection points preceding $\sigma_{\Delta,x}$ and following $\sigma_{\Delta,w}$ in Σ^{-1} must also be incident on Δ. Removing γ from Σ^{-1} gives $\Sigma^{-1} = (\Sigma_n, \cdots, \sigma_{\Delta,y}, \sigma_{\Delta,z}, \cdots, \Sigma_1)$. This is now a simple path where every disc occurs at twice and is adjacent. The results of Proposition 2.2.3 can then be applied to obtain the remaining two new regions.

Generally, if there are $m > 0$ such cycles in Σ the regions they define are computed accordingly. Let $\mathcal{M}_k$ $(k \leq m)$ be the set of connection points of the kth cycle of Σ. The regions created by these cycles is given by $\mathcal{M}_k^{-1}$, as explained above. The remaining two regions defined by this new embedding are therefore given by the following:

$$\begin{aligned} \rho_{i,m+1} &= (\sigma_{\Delta_x,a}, \Sigma, \sigma_{\Delta_y,h}, \xi_2) \\ \rho_{i,m+2} &= (\xi_1, \sigma_{\Delta_y,f}, \Sigma^{-1} - \sum_{k=1}^{m} \mathcal{M}_k, \sigma_{\Delta_x,c}) \end{aligned}$$

Note that if Σ does not contain any cycles, $m = 0$ and the region $\rho_{i,2}$ is given by $(\xi_1, \sigma_{\Delta_y,f}, \Sigma^{-1}, \sigma_{\Delta_x,c})$, which is in agreement with the results of Proposition 2.2.3. □

2.3 An Example to Illustrate the Planarity Testing Algorithm

This algorithm is illustrated by means of an example. Consider the graph given in Figure 2.15

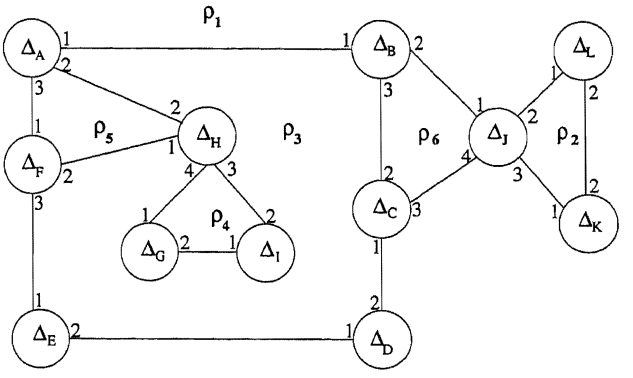


Figure 2.15: A graph to illustrate the algorithm

The initial cycle, C, of Γ is calculated by finding a connection point, $\sigma_{\Delta_A,1}$ say, in Γ, traversing the arc $(\sigma_{\Delta_A,1}, \sigma_{\Delta_B,1})$, the corner $(\sigma_{\Delta_B,1}, \sigma_{\Delta_B,2})$ in a clockwise direction, the arc $(\sigma_{\Delta_B,2}, \sigma_{\Delta_J,1})$ and so on until a cycle of Γ has been encountered. The first cycle encountered is:

$$C = (\sigma_{\Delta_J,2}, \sigma_{\Delta_L,1}, \sigma_{\Delta_L,2}, \sigma_{\Delta_K,2}, \sigma_{\Delta_K,1}, \sigma_{\Delta_J,3})$$

Embedding C in space $\mathcal{S}$ gives the graph Γ'. Applying the results given in Lemma 2.2.1 gives the following regions of Γ':

$$\rho_1 = C,$$
$$\rho_2 = C^{-1}$$

The regions ρ_1 and ρ_2 in Γ' are depicted in Figure 2.16.

Both Γ and Γ' contain the disc Δ_J. This disc has a connection point, $\sigma_{\Delta_J,4}$, that is embedded in Γ and not Γ'. This connection point becomes the first element in the path Σ. This path is constructed until a connection point is encountered that is incident on a disc associated

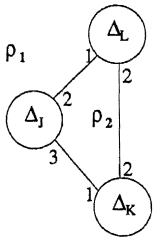


Figure 2.16: The regions of Γ' defined by C

in Γ'. The path Σ is therefore:

$$\begin{aligned}\Sigma = (&\sigma_{\Delta_J,4}, \sigma_{\Delta_C,3}, \sigma_{\Delta_C,1}, \sigma_{\Delta_D,2}, \sigma_{\Delta_D,1},\\ &\sigma_{\Delta_E,2}, \sigma_{\Delta_E,1}, \sigma_{\Delta_F,3}, \sigma_{\Delta_F,1}, \sigma_{\Delta_A,3},\\ &\sigma_{\Delta_A,1}, \sigma_{\Delta_B,1}, \sigma_{\Delta_B,2}, \sigma_{\Delta_J,1})\end{aligned}$$

Now, $\sigma_{\Delta_J,4}$ can be embedded in ρ_1 in the corner given by $(\sigma_{\Delta_J,3}, \sigma_{\Delta_J,2})$. Similarly $\sigma_{\Delta_J,1}$ can also be embedded into ρ_1 in this corner. As the first and last connection points of Σ can be embedded in the same region of Γ', the path Σ can be embedded in Γ'.

The path Σ is a simple path. The new regions of Γ' created by this embedding are therefore determined by using the results in Proposition 2.2.3:

$$\begin{aligned}\rho_{1,1} = (&\sigma_{\Delta_J,3}, \sigma_{\Delta_J,4}, \sigma_{\Delta_C,3}, \sigma_{\Delta_C,1}, \sigma_{\Delta_D,2},\\ &\sigma_{\Delta_D,1}, \sigma_{\Delta_E,2}, \sigma_{\Delta_E,1}, \sigma_{\Delta_F,3}, \sigma_{\Delta_F,1},\\ &\sigma_{\Delta_A,3}, \sigma_{\Delta_A,1}, \sigma_{\Delta_B,1}, \sigma_{\Delta_B,2}, \sigma_{\Delta_J,1},\\ &\sigma_{\Delta_J,2}, \sigma_{\Delta_L,1}, \sigma_{\Delta_L,2}, \sigma_{\Delta_K,2}, \sigma_{\Delta_K,1},)\\ \rho_{1,2} = (&\sigma_{\Delta_B,2}, \sigma_{\Delta_B,1}, \sigma_{\Delta_A,1}, \sigma_{\Delta_A,3}, \sigma_{\Delta_F,1},\\ &\sigma_{\Delta_F,3}, \sigma_{\Delta_E,1}, \sigma_{\Delta_E,2}, \sigma_{\Delta_D,1}, \sigma_{\Delta_D,2},\\ &\sigma_{\Delta_C,1}, \sigma_{\Delta_C,3}, \sigma_{\Delta_J,4}, \sigma_{\Delta_J,1})\end{aligned}$$

For simplicity these regions of Γ' are renamed: $\rho_{1,1}$ becomes ρ_1 and $\rho_{1,2}$ becomes ρ_3. The graph Γ' is shown in Figure 2.17.

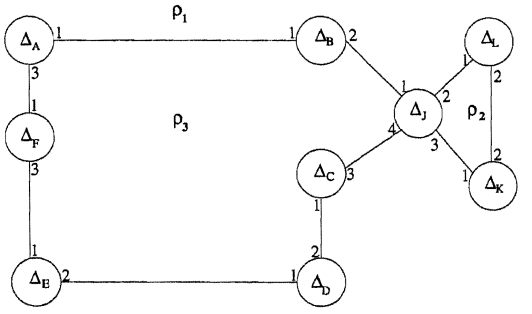


Figure 2.17: The regions of Γ' after embedding Σ

By applying the algorithm again, a new path Σ is constructed:

$$\begin{aligned}\Sigma \;=\; & (\sigma_{\Delta_A,2}, \sigma_{\Delta_H,2}, \sigma_{\Delta_H,3}, \sigma_{\Delta_I,2}, \sigma_{\Delta_I,1}, \\ & \sigma_{\Delta_G,2}, \sigma_{\Delta_G,1}, \sigma_{\Delta_H,4}, \sigma_{\Delta_H,1}, \sigma_{\Delta_F,2})\end{aligned}$$

As can be seen, the first and last connection points of Σ, $\sigma_{\Delta_A,2}$ and $\sigma_{\Delta_F,2}$ can both be embedded in region ρ_3 of Γ'. This path Σ can therefore be embedded without violating planarity of Γ'. There is one cycle in Σ; applying the results of Proposition 2.2.4 gives the following region of Γ':

$$\rho_{3,1} \;=\; (\sigma_{\Delta_H,4}, \sigma_{\Delta_G,1}, \sigma_{\Delta_G,2}, \sigma_{\Delta_I,1}, \sigma_{\Delta_I,2}, \sigma_{\Delta_H,3})$$

The remaining two regions formed by this embedding are given by:

$$\begin{aligned}\rho_{3,2} \;=\; & (\sigma_{\Delta_A,1}, \sigma_{\Delta_A,2}, \sigma_{\Delta_H,2}, \sigma_{\Delta_H,3}, \sigma_{\Delta_I,2}, \sigma_{\Delta_I,1}, \\ & \sigma_{\Delta_G,2}, \sigma_{\Delta_G,1}, \sigma_{\Delta_H,4}, \sigma_{\Delta_H,1}, \sigma_{\Delta_F,2}, \sigma_{\Delta_F,3}, \\ & \sigma_{\Delta_E,1}, \sigma_{\Delta_E,2}, \sigma_{\Delta_D,1}, \sigma_{\Delta_D,2}, \sigma_{\Delta_C,1}, \sigma_{\Delta_C,2}, \\ & \sigma_{\Delta_B,3}, \sigma_{\Delta_B,1}) \\ \rho_{3,3} \;=\; & (\sigma_{\Delta_A,3}, \sigma_{\Delta_F,1}, \sigma_{\Delta_F,2}, \sigma_{\Delta_H,1}, \sigma_{\Delta_H,2}, \sigma_{\Delta_A,2})\end{aligned}$$

Again these region regions are renamed ρ_4, ρ_3 and ρ_5 respectively and are shown in Figure 2.18.

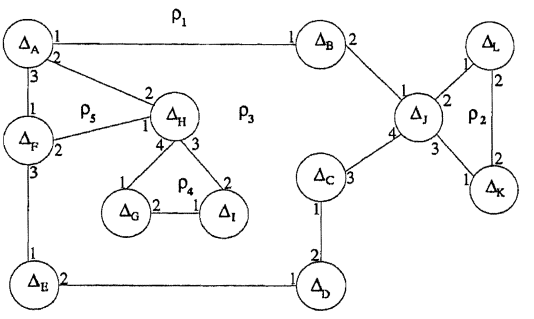


Figure 2.18: The regions of Γ' after embedding Σ

There are now only two connection points of Γ that have not been embedded in Γ'. These connection points form the path $(\sigma_{\Delta_B,3}, \sigma_{\Delta_C,2})$. This simple path can be embedded into region ρ_3. This region is now characterised as follows:

$$\begin{aligned}\rho_3 \;=\; & (\sigma_{\Delta_A,1}, \sigma_{\Delta_A,2}, \sigma_{\Delta_H,2}, \sigma_{\Delta_H,3}, \sigma_{\Delta_I,2}, \sigma_{\Delta_I,1},\\ & \sigma_{\Delta_G,2}, \sigma_{\Delta_G,1}, \sigma_{\Delta_H,4}, \sigma_{\Delta_H,1}, \sigma_{\Delta_F,2}, \sigma_{\Delta_F,3},\\ & \sigma_{\Delta_E,1}, \sigma_{\Delta_E,2}, \sigma_{\Delta_D,1}, \sigma_{\Delta_D,2}, \sigma_{\Delta_C,1}, \sigma_{\Delta_C,2},\\ & \sigma_{\Delta_B,3}, \sigma_{\Delta_B,1})\end{aligned}$$

The remaining new region is $\rho_6 = (\sigma_{\Delta_B,2}, \sigma_{\Delta_B,3}, \sigma_{\Delta_C,2}, \sigma_{\Delta_C,3}, \sigma_{\Delta_J,4}, \sigma_{\Delta_J,1})$. Now Γ' is identically equal to the graph, Γ, given in Figure 2.15 and it can therefore be concluded that the rotation scheme associated with Γ depicts a planar graph.

2.4 Implementation, Experiments and Conclusions

The planarity testing algorithm was implemented on GAP in the program `PlanarityTest`. The algorithm takes the adjacency lists representing the rotation scheme of a, possibly non-simple, graph Γ as input. When Γ proves to be planar, the algorithm returns all regions of Γ. If a path of Γ can not be embedded in a topological space without violating planarity then the algorithm stops and produces a message to this effect. If Γ contains more than one bi-connected component a function, `PlanarityTestWithRegions`, is automatically called. This function is an adaption of `PlanarityTest` and computes the regions of each bi-connected

component in turn.

The program was tested on equal numbers of planar and non-planar graphs with the number of vertices in these tests ranging from 2-100, arcs from 2-290. For these tests, performed on an Intel Pentium 2 machine using Red Hat Linux 6 and version 4.2 of GAP, the maximum computation time of the algorithm for planar graphs was $3.1 * 10^{-1}$ seconds and $1 * 10^{-2}$ seconds for non-planar graphs.

In conclusion, this algorithm determines planarity by examining the rotation scheme of an arbitrary graph in a suitable time frame. In order to achieve this, both planar and non-planar graphs are represented by rotation schemes. Preceding this work, rotation schemes have only been used to characterise the structure of planar graphs. One of the main advantages of this planarity testing technique is that the regions of the graph are computed simultaneously to planarity being determined. Previous algorithms (see [25], [43]) that extract regions from a given graph require that the planarity of the graph is determined beforehand. In relation to the work presented in this thesis, extracting the regions in this way is particularly useful for the algorithms given in Chapters 3 and 5.

An important feature of this algorithm is that it works for non-simple graphs. Previous research in this area concentrates on simple graphs. The algorithm itself has particular applications in such areas as genetics, communication design, VLSI and network optimisation; disciplines where the order of the arcs incident on discs could be important in the representation of the system. This is obviously true for spherical pictures, and rotation schemes are a very useful tool to represent such graphs. In this chapter an efficient data structure for pictures has been presented together with methods to verify that this data structure does indeed represent a picture over a group presentation.

Chapter 3

Determining Geometric and Algebraic Definitions of π_2

Let $\mathcal{P} = < \mathbf{x}; \mathbf{r} >$ be a presentation for a group G. A *3-presentation* for G is given by $< \mathcal{P}; \mathbf{s} >$; where $\mathbf{s}$ denotes the generators of the second homotopy module, $\pi_2(\mathcal{P})$. There are several ways to describe $\mathbf{s}$: geometrically, by means of a collection of spherical pictures; algebraically, by a set of identity sequences (also referred to in the literature as identities among relations); and topologically. It is the first two of these methods that this chapter is concerned with. A technique is given that produces a collection of spherical pictures over $\mathcal{P}$ from an algebraic form for $\mathbf{s}$. Conversely, if $\mathbf{s}$ is given in a geometric form, a technique is provided that computes the identity sequences that describe $\pi_2(\mathcal{P})$. There are computational packages (see [24] and [34]) that characterise π_2 algebraically, but currently there are no methods that produce a geometric description. Before these algorithms are presented, a summary of both of the aforementioned computational packages is given.

3.1 Existing Computational Packages

The work of Ellis and Kholodna [24] computes a *3-presentation* for every group of order at most 30. Recall from the algebraic definition of a *3-presentation* (see Page 13) that $\mathbf{s}$ is a

set of identity sequences such that the set

$$\{< \sigma >: \sigma \in \mathbf{s}\}$$

generates the module of identity sequences. Ellis and Kholodna's [24] method for computing such a set $\mathbf{s}$ of identity sequences is based upon finding the kernel of the homomorphism of:

$$\xi : \oplus_{r \in \mathbf{r}} \mathbf{Z}G \to \oplus_{x \in \mathbf{x}} \mathbf{Z}G.$$

This homomorphism is defined on basis elements in terms of the Whitehead-Reidemeister-Fox derivatives. Determining $\ker \xi$ is achieved by first computing a $\mathbf{Z}$-basis for $\ker \xi$, and then trying to find a minimal subset, $\mathbf{s}$, of this which generates $\ker \xi$ as a $\mathbf{Z}G$-module. This set $\mathbf{s}$ cannot be computed automatically at present and the authors use Cayley graphs of the presentation and "human ingenuity" to find such a set of elements.

The technique of Heyworth and Wensley [34] uses a logged rewrite system, $\mathcal{L}(\mathcal{P})$, of a group presentation $\mathcal{P} = < \mathbf{x}; \mathbf{r} >$ to the associated monoid presentation $\mathcal{P}'$, to identify a generating set of identity sequences. Logged re-write systems are procedures which store all the operations performed on $\mathcal{P}$ to obtain $\mathcal{P}'$. Brown and Razak Salleh [8] show how to obtain a generating set for the second homotopy module of $\mathcal{P}'$ from a morphism, Υ, from the free groupoid on the Cayley graph of $\mathcal{P}$ to the free crossed module on $\mathcal{P}$. The program of Heyworth and Wensley is based upon this result: it uses logged rewrite systems to find such an Υ, and from this they use logged reduction methods to find a set of identities among relations in $\mathcal{P}$. In a sequel to their work, [33], it is shown how logged rewrite systems can be used to find a minimal generating set for $\pi_2(\mathcal{P})$.

3.2 Algorithm to Determine a Representation of a Picture From an Identity Sequence

Given an identity sequence, θ of $\mathcal{P} =< \mathbf{x}; \mathbf{r} >$, the aim is to obtain a representation of the spherical picture, $\mathbf{P}$ say, as defined by θ. The representation of the picture will be given by lists storing the rotation scheme[1], `RotationScheme`, and the relator information, `RelatorInformation`, for $\mathbf{P}$, as detailed in Chapter 2. Lists are also used to store identity sequences. Every word of the form $W_i R_i^{\epsilon_i} W_i^{-1}$ (W_i a word on $\mathbf{x}$, $R_i^{\epsilon_i} \in \mathbf{r}$) in a sequence is stored in a list of the form: $[R_i^{\epsilon_i}, W_i]$. It is then simply a matter of multiplying components in such a list of lists to obtain the desired sequence.

Example 1.1 (ctd.)

The list for the identity sequence given in (1.3) is therefore:

$$\theta = [\ [T^{-1}, b], [S, 1], [R, b^{-1}], [S, (ab)^{-1}], [R, (bab)^{-1}], [S, (abab)^{-1}],$$
$$[R, (babab)^{-1}], [T^{-1}, (baba)^{-1}]\ ],$$

where 1 denotes the empty word.

Using this formula, θ is represented in a list, $\mathcal{L}$. Determining `RelatorInformation` from θ is a straightforward procedure. Obviously each component of $\mathcal{L}$ corresponds to a disc in $\mathbf{P}$. A value is assigned to each disc in $\mathbf{P}$, based on the position of its relator in $\mathcal{L}$. Given the jth component of $\mathcal{L}$ ($1 \leq j \leq |\mathcal{L}|$), $[R_j^{\epsilon_j}, W_j]$ say, the number j is added to either the list `PositiveDiscs` or the `NegativeDiscs` list of the record for R_j, depending on the value of ϵ_j. The value for `Relator` is equal to R_j. Once this procedure has been carried out for every component of $\mathcal{L}$ the complete list `RelatorInformation` for $\mathbf{P}$ is obtained, with the exception of the component `Label`. This is because it is difficult to generate labels automatically. In any case these labels are not required for this algorithm and, if required, they can be easily user-defined after the algorithm has executed.

[1] For clarity, from this point on, the list for the rotation scheme of a picture will be called `RotationScheme` rather than A.

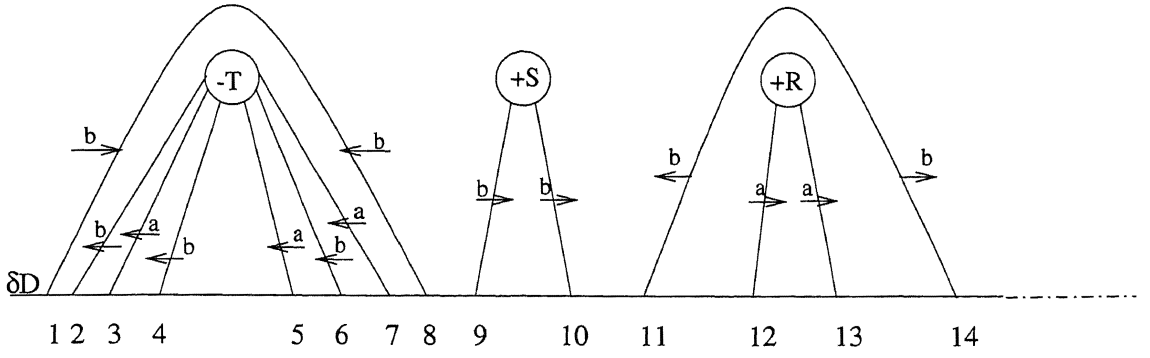


Figure 3.1: A visualisation of part of the sequence given by (1.3)

Example 1.1 (ctd.)

The relator information corresponding to the sequence given in (1.3) is therefore:

$$\begin{aligned}\mathcal{L} = \quad [\quad & \texttt{Relator} := a^2, \texttt{PositiveDiscs} := [3,5,7], \texttt{NegativeDiscs} := [0]; \\ & \texttt{Relator} := b^2, \texttt{PositiveDiscs} := [2,4,6], \texttt{NegativeDiscs} := [0]; \\ & \texttt{Relator} := (ab)^3, \texttt{PositiveDiscs} := [0], \texttt{NegativeDiscs} := [1,8] \quad]\end{aligned}$$

The algorithm then proceeds by determining the rotation scheme for $\mathbf{P}$. Obtaining a spherical picture from a given identity sequence by hand involves placing $\mathbf{x}$-arcs, given by the relators of θ, and free arcs, given by words of the form W^ϵ in θ, on a boundary, ∂D. Figure 3.1 shows such a diagram for the first three terms of the identity sequence given by (1.3). The numbers on the boundary denote the position of the arcs on ∂D. For instance, position 1 in Figure 3.1 depicts the location, on the boundary, of the first endpoint of the first free arc. Note that these numbers are not required when obtaining the picture by hand, but are given here to aid explanation.

The arrangement of arcs on the boundary is given by their corresponding position in θ. The corresponding picture is then obtained by cancelling adjacent arcs if their labels have the form $x^\epsilon x^{-\epsilon}$, for $x \in \mathbf{x}, \epsilon = \pm 1$. For example, in Figure 3.1 the free arc at position 8 on the boundary can cancel with the $\mathbf{x}$-arc at position 9 since the labels of these arcs give $b^{-1}b$ on the boundary ∂D. Figure 3.2 shows the resulting diagram after all cancelling operations have been performed on Figure 3.1. Drawing arcs on a boundary together with

this cancelling procedure can be quite complex and time consuming, especially in the case of sequences with many elements. It is therefore useful to produce an automatic method for obtaining pictures from identity sequences.

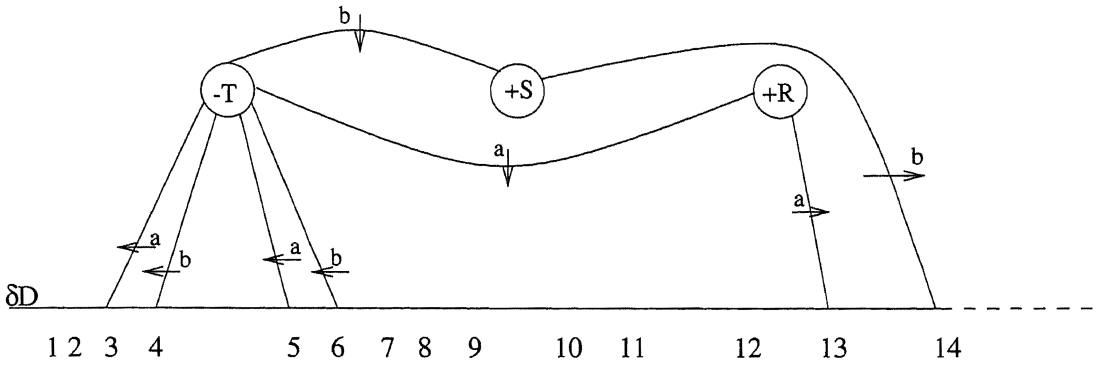


Figure 3.2: Figure 3.1 after all cancelling operations have been performed

The technique adopted to obtain the rotation scheme of **P** involves constructing a list, `Boundary`, which stores the position of arcs on ∂D together with their labels. If the arc is a free arc, its corresponding component in `Boundary` has 3 parts: the number of its start and end position on ∂D and the label of the free arc, with respect to its start position. For **x**-arcs, which are connected to discs, the corresponding component in `Boundary` stores the position of the arc on ∂D together with an additional list of length 2. This list gives the disc that the arc is connected to and the position of the arc incident upon that disc. The reason that **x**-arcs contain a list within a list is so they can be distinguished easily (computationally) from free arcs.

Example 1.1 (ctd.)

The list `Boundary` for the picture given in Figure 3.1 is therefore:

$$\begin{aligned}[\quad & [1,8,b],[2,[1,1]],[3,[1,2]],[4,[1,3]],[5,[1,4]],[6,[1,5]], \\ & [7,[1,6]],[9,[2,1]],[10,[2,2]],[11,14,b^{-1}],[12,[3,1]],[13,[3,2]] \ ]\end{aligned} \tag{3.1}$$

This list is calculated by putting the first component of $\mathcal{L}$, $[T^{-1},b]$, into the form $\mathcal{L}_1 = bT^{-1}b^{-1}$. (Recall that $\mathcal{L} = [\mathcal{L}_1,\mathcal{L}_2,\cdots,\mathcal{L}_{|\Delta|}]$, where Δ is the collection of discs in **P**.) As $T^{-1} = (ab)^{-3}$, $\mathcal{L}_1$ therefore has length 8. The free arcs of $\mathcal{L}_1$ are inspected first.

The first component of Boundary is calculated by taking the positions of the first and last elements of $\mathcal{L}_1$ together with the label of the element at the first position, i.e. $[1, 8, b]$, as required. If the element at the second position on the boundary is a free arc, the start and end positions of this arc and its label are stored at the next component in Boundary. This procedure continues until an **x**-arc is reached; in this example the first **x**-arc on the boundary is the first element of T^{-1}, b^{-1}. The position of this arc on the boundary, two in this case, is saved together with the disc number that the arc is incident upon and the position of this arc on the disc. The disc's index corresponds to the component of $\mathcal{L}$ that is being examined, in this case this gives $[1, 1]$. The corresponding entry for this **x**-arc in Boundary is therefore $[2, [1, 1]]$. Once all elements in $\mathcal{L}_1$ have been placed in Boundary, this method is then repeated for successive components of $\mathcal{L}$.

The next stage of the algorithm is to cancel arcs to form the rotation scheme of **P**. To do this two counters are initialised. The left counter, **CLeft**, is set to position 1 in Boundary, whilst the right counter, **CRight**, is initialised to position 2. The concept of the algorithm is to move these counters along the boundary until the labels of the arcs or free arcs at their position on the boundary cancel.

The following four possibilities arise:

1. A free arc can cancel with a free arc. Let the entry for **CLeft** in Boundary be $[a, A, x]$ and that of **CRight** be $[b, B, x^{-1}]$, where $a < b < B < A$, then a new free arc is formed and one of the following occur:

   - Both **CLeft** and **CRight** depict the first position of the free arcs on the boundary. The corresponding entry for **CLeft** in Boundary is updated to store the second position of the free arcs at the counters of **CLeft** and **CRight** together with the label of the free arc at **CLeft**. The arc at the corresponding position for **CRight** in Boundary is removed from Boundary. Therefore, as shown in Figure 3.3, the entry of **CLeft** in Boundary changes from $[a, A, x]$ to $[B, A, x]$ and the entry for **CRight**, $[b, B, x^{-1}]$ is removed.

   - Both **CLeft** and **CRight** depict the second position of the free arcs on the

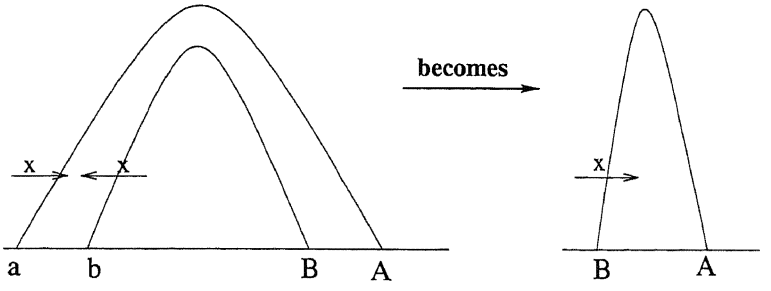


Figure 3.3: An example of cancelling free arcs

boundary. Here the entry for **CLeft** in Boundary becomes $[a, b, x]$ and that for **CRight** is deleted.

- Let $[a, A, x]$, $[b, B, x]$ and $a < A < b < B$, which gives the situation depicted in Figure 3.4. Let **CLeft** denote the arc at position A on the boundary and **CRight**

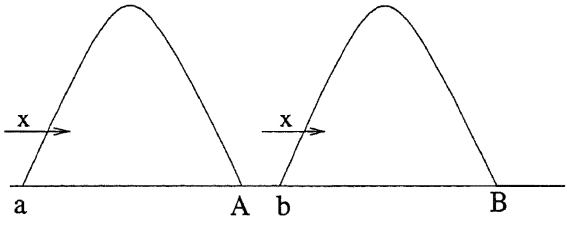


Figure 3.4: An example of cancelling free arcs

is situated at position b. The entry $[a, A, x]$ is changed to become $[a, B, x]$ and $[b, B, x^-1]$ is removed from Boundary.

2. A free arc, denoted $[a, A, x]$ in Boundary can cancel with an **x**-arc, represented by $[b, [R, j]]$ for an arbitrary relator R say. If this is the case then a new arc is formed, the entry for **CRight** is removed from Boundary and one of the following occurs:

   - If $a < b < A$ and **CLeft** is located at position a, then the entry for **CLeft** in Boundary becomes $[A, [R, j]]$. This possibility is depicted in Figure 3.5. Similarly, if $a < A < b$ and **CLeft** lies at A on the boundary then $[a, A, x]$ becomes $[a, [R, j]]$.
   - **CLeft** depicts the **x**-arc at position b on the boundary. If $b < a < A$ then this arc is represented by $[A, [R, j]]$. Alternatively, if $a < b < A$ then **CLeft** becomes $[a, [R, j]]$.

3. An arc can cancel with another arc. If this is the case then there is an arc which does not lie on the boundary and is therefore an arc in **P**. The details of this arc are then

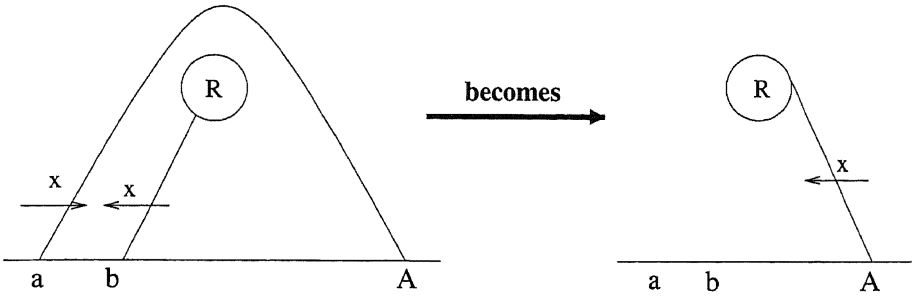


Figure 3.5: An example of cancelling a free arc with an arc

added to the list `RotationScheme` by storing the discs, arc numbers and labels and then both arcs are removed from `Boundary`.

4. No cancellation occurs. In this case the values of **CLeft** and **CRight** are incremented by 1.

If any of the possibilities (1)-(3) given above occur then **CLeft** is decreased by 1 whilst **CRight** is increased by 1. This procedure continues until `Boundary` is empty or **CRight** is greater than the number of elements on the boundary. This latter situation implies that θ, represented by $\mathcal{L}$, is not an identity sequence as there exist elements on the boundary that cannot cancel. If `Boundary` does become empty however, θ is an identity sequence and its spherical picture is represented by `RotationScheme` and `RelatorInformation`.

Example 1.1 (ctd.)

Implementing this final stage of the algorithm on the list given in (3.1) gives the

Position of **CLeft**	Position of **CRight**	Action
1	2	(2) occurs. Replace $[1, 8, b]$ with $[8, [1, 1]]$ Set **CLeft**=2 and **CRight**=3.
8	9	(3) occurs. Set `RotationScheme`[1][1] = $[2, 1, b^{-1}]$ and `RotationScheme`[2][1] = $[1, 1, b]$. Set **CLeft**=7 and **CRight**=10
10	11	(2) occurs. Replace $[10, [2, 2]]$ with $[14, [2, 2]]$ Set **CLeft**=7 and **CRight**=12
7	12	(3) occurs. Set `RotationScheme`[1][7] = $[3, 1, a^{-1}]$ and `RotationScheme`[3][1] = $[1, 7, a]$.

Table 3.1: A table to show the computational stages of the algorithm

calculations detailed in Table 3.1 (the stages where possibility (4) occurs and the delet-

ing of entries from `Boundary` are omitted for brevity).

3.3 Determining Identity Sequences from Pictures

Let $\mathbf{P}$ be a spherical picture over a presentation $\mathcal{P}$ that is represented by a rotation scheme and a list `RelatorInformation`. In order to compute the identity sequence associated with $\mathbf{P}$, it is necessary to determine the sequence in the form of (1.2), where $\gamma = (\gamma_1, \gamma_2, \cdots, \gamma_n)$ denotes a spray for $\mathbf{P}$. Due to the fact that the relator associated with every disc in $\mathbf{P}$ is given in `RelatorInformation`, this essentially involves computing the paths, γ_i, and identifying the labels, $W(\gamma_i)$, for every $\gamma_i \in \gamma$ and the order in which they occur around the point O.

The first stage of the algorithm to compute such a sequence from a picture, is to verify that the representation of $\mathbf{P}$ does indeed depict a picture over $\mathcal{P}$. This involves calling the algorithms described in Chapter 2: verifying that the word associated with every disc is a relator of $\mathcal{P}$, and that $\mathbf{P}$ is a planar graph. If $\mathbf{P}$ is a planar graph then the regions of $\mathbf{P}$ are produced. Recall that the corners of discs in each region are traversed in a clockwise direction which, in turn, gives an anti-clockwise orientation for regions. The connection point of each region is given in the form $[\Delta_i, j, W(\sigma_{\Delta_i,j})]$, where j is the number of the arc incident on disc Δ_i and $W(\sigma_{\Delta_i,j})$ denotes the label of this connection point, again with regards to the clockwise orientation of Δ_i.

Example 1.1 (cont.)

A region of a picture of S_3 given in Figure 1.12 is redrawn in Figure 3.6 with numbers assigned to each arc endpoint. This region would then be stored as:

$$[\ [T^{-1}, 3, b^{-1}], [S, 3, b], [S, 1, b], [T^{-1}, 5, b^{-1}],$$
$$[T^{-1}, 6, a^{-1}], [R, 2, a], [R, 1, a], [T^{-1}, 2, a^{-1}]]$$

The fact that the regions of the picture are extracted at the same time that planarity is determined is extremely useful for this algorithm. The function `CheckDiscs`, used to verify that the word associated with each disc is a relator of the presentation, also serves more

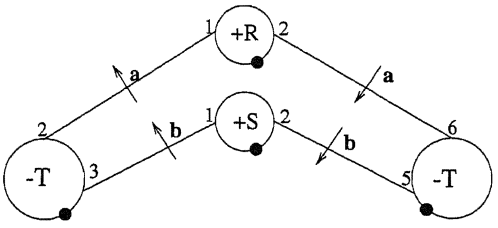


Figure 3.6: A region of the picture given in Figure 1.12

than one purpose. Once it is known that a word associated with a disc equals a relator, the function can also be used to give a possible location of the basepoint of that disc.

Let n be the length of the relator, R, of $\mathcal{P}$, represented by Δ in $\mathbf{P}$. `CheckDiscs` reads the labels of the arcs emanating from Δ and determines a word $W(\sigma_{\Delta,i})W(\sigma_{\Delta,i+1})\cdots W(\sigma_{\Delta,i-1})$, for $1 \leq i \leq n \pmod{deg(\Delta)}$, which is identically equal to R. Hence, if Δ has a positive orientation, a basepoint of Δ lies in the region that contains the components $[\Delta, i-1, W(\sigma_{\Delta,i-1})], [\Delta, i, W(\sigma_{\Delta,i})]$. For negatively oriented Δ, a basepoint lies in the region that contains the components $[\Delta, i, W(\sigma_{\Delta,i})], [\Delta, i+1, W(\sigma_{\Delta,i+1})]$. These examples are illustrated in Figure 3.7. The procedure `DiscDetails` calls `CheckDiscs` and from the result of this function, `DiscDetails` produces a list for all discs in $\mathbf{P}$. This list stores the relator the disc represents, the direction of the disc and the region where a basepoint of this disc lies.

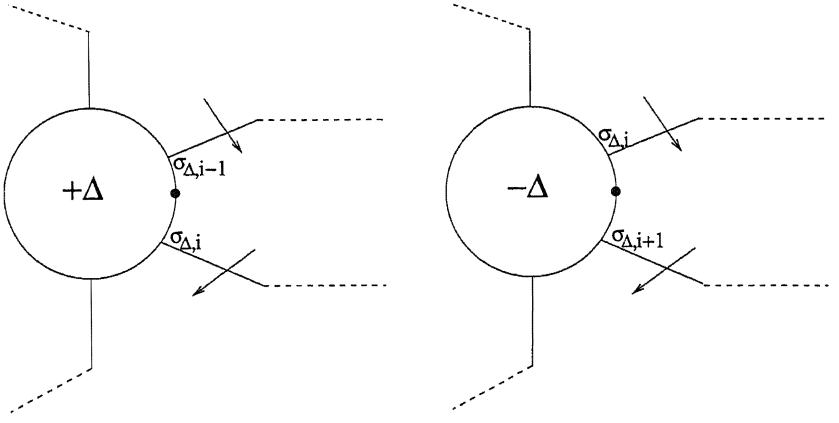


Figure 3.7: Determining locations of basepoints

Obviously there could be more than one corner of a disc where a basepoint could lie. These different locations would alter the sequence obtained from the picture. The aim of this algorithm, however, is to produce a valid sequence for the given picture. It would not be a difficult matter to obtain as many sequences as the picture could represent. All that is

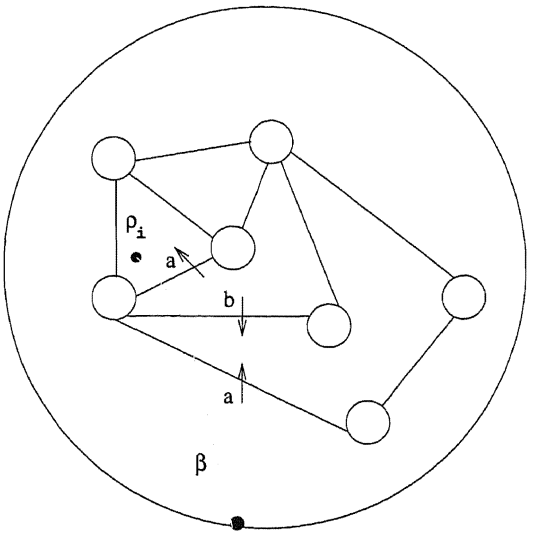


Figure 3.8: A path β from O to region ρ_1

required is to iteratively call the `DiscDetails` function to obtain all possible regions where the basepoints of each disc could lie. It can be proved, [64], however that any two sequences obtained from the same picture are Peiffer equivalent. It is therefore enough to find one such sequence.

In order to determine an identity sequence associated with $\mathbf{P}$, it is necessary to determine the labels of the paths from a point O, on the boundary of $\partial\mathbf{P}$, to the basepoint of each disc in $\mathbf{P}$. This is achieved by computing the word from O to each region in $\mathbf{P}$. Given a path β, where β has an initial point O and terminal point in the interior of a region, the *word associated with a region* is the term employed to describe the word constructed by reading the values of the arcs encountered on a complete walk of β. For instance, a word associated with region ρ_i in Figure 3.8 is $ab^{-1}a$. Given a word w_j associated with region ρ_j, say, the labels of the paths in the spray that terminate at basepoints situated in ρ_j can be determined from the word w_j and the relators of the discs whose basepoints lie in ρ_j. Applying this method to all regions in $\mathbf{P}$ gives the labels of all the paths in the spray.

The algorithm therefore proceeds by determining the word associated every region of $\mathbf{P}$. Evidently, the word associated with the exterior region of $\mathbf{P}$ is equal to the empty word. As $\mathbf{P}$ can be manipulated in such a way that any of it's regions can become the exterior region; for consistency, the exterior region is chosen to be the region with the most discs. Note,

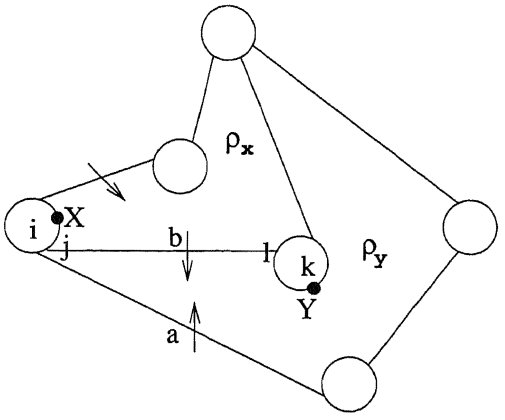


Figure 3.9: A diagram to show the label needed to access ρ_y from ρ_x

however, that the choice of exterior region also effects which sequences are produced. The chosen exterior region, ρ_1 say, is stored together with the identity of $\mathcal{P}$ in a list `RegionLabel`.

An arc of $\mathbf{P}$ bounds two regions where, for each of these two regions, the order of the connection points of that arc is reversed. That is, if $\{\sigma_{i,j}, \sigma_{k,l}\}$ is an arc in a region of $\mathbf{P}$, then the arc must also be contained in another region of $\mathbf{P}$ where it is ordered $\{\sigma_{k,l}, \sigma_{i,j}\}$. The way the word associated with each region is determined is given by the following lemma:

Lemma 3.3.1 *Let $\{\sigma_{i,j}, \sigma_{k,l}\}$ be an arc in a region, ρ_x say, of a picture $\mathbf{P}$. The label of this arc is $W(\sigma_{i,j})$ as traversed clockwise around the disc Δ_i. Let $\{\sigma_{k,l}, \sigma_{i,j}\}$ be an arc in region ρ_y. Hence, one way in which ρ_y can be entered directly from ρ_x is by crossing the arc $\{\sigma_{i,j}, \sigma_{k,l}\}$. If the word associated with ρ_x is known to be w say, then the word associated with ρ_y becomes $w \cdot W(\sigma_{i,j})$.*

Proof It is obvious that if two regions are bounded by an arc, one of these regions can be accessed from the other by crossing this arc. The label of this crossing is either given by $W(\sigma_{i,j})$ or $W(\sigma_{k,l})$. (Note that $W(\sigma_{i,j}) = W(\sigma_{k,l})^{-1}$.)

Recall that the labels of connection points are given with respect to a clockwise orientation around each disc. Due to this fact, in Figure 3.9 traversing the disc Δ_k in a clockwise direction from the point Y and crossing the arc takes one from ρ_y into ρ_x, however traversing Δ_i from a point X and crossing the arc takes one from ρ_x into ρ_y, as required. The label required is therefore the label of the first connection point of the arc in the region that one

is entering ρ_y from. This result always holds due to the fact that discs are traversed in a clockwise direction. Travelling (clockwise) over the jth connection point of Δ_i enables one to enter into the desired region; whereas the only way in which $\sigma_{k,l}$ could be crossed to travel from ρ_x into ρ_y is in an anti-clockwise direction, which violates how regions are constructed. The label needed to access ρ_y from ρ_x is therefore given by $W(\sigma_{i,k})$. Evidently the label of the path from O to the interior of ρ_y is equal to the label of the path from O into ρ_x concatenated with $W(\sigma_{i,j})$. □

Once the word associated with region has been computed it is stored together with that region's index in a list `RegionLabel`. If a region, ρ_z say, does not have a word associated with it, the arcs of the regions whose index is contained in `RegionLabel` are searched through until an arc is found that is also in ρ_z. If no such arc exists, then the arcs of another region are examined. This procedure is continued until all the regions of $\mathbf{P}$ have a word associated with them. This can always be achieved as a picture is a connected graph. The label of every path in γ can then be determined.

Recall that the paths in the spray are not allowed to cross and that the identity sequence obtained from $\mathbf{P}$ is given by reading the labels of the paths in the spray in a clockwise order around O. The last stage of the algorithm is therefore to put these paths in the correct order to produce an identity sequence. This involves storing the manner in which the words associated with regions were formed. The arc which enables region ρ_y is entered into from region ρ_x is said to be an *access arc* from ρ_x.

Example 3.1

Arcs[2] $\{3,4\}$ and $\{4,6\}$ are the access arcs from ρ_2 in Figure 3.10.

The list `AccessArcs` is created concurrently to `RegionLabel`. If the word associated with ρ_y is determined by crossing the arc that bounds ρ_x and ρ_y, then this arc and the index y is appended to the xth component of `AccessArcs`. The xth component of `AccessArcs` thus gives a list of all access arcs and the regions that they enter into from region ρ_x.

[2]Note that for convenience the connection points of the picture in Figure 3.10 have not been labelled and are not referred to in Example 3.1. In practice, the arcs and regions are computed in the standard way, with connection points.

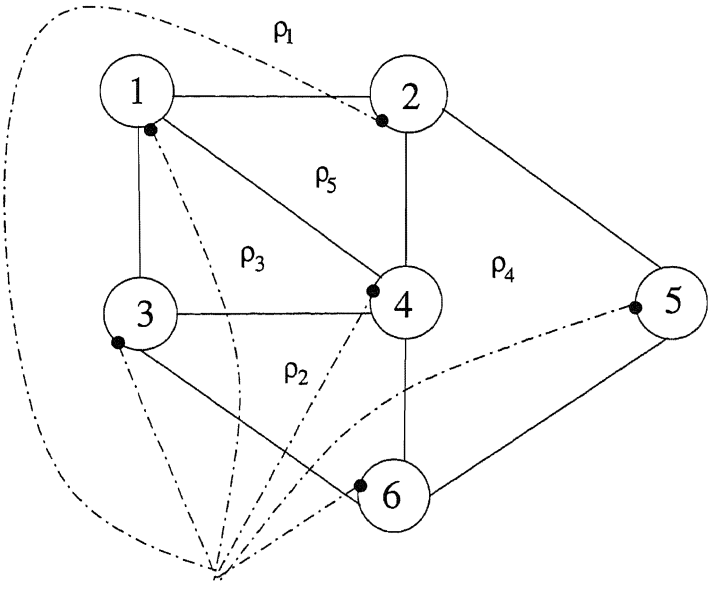


Figure 3.10: A diagram to show the search routes of a picture

Once all access arcs of **P** have been computed, the connection points of each region are now reordered such that the corners of the regions have an anti-clockwise orientation. The first connection point of the region is the first connection point of the access arc into that region. An access arc from the exterior region into a boundary region is chosen to reorder the connection points of the exterior region. The choice of this access arc is immaterial. Changing the direction of corners means that regions have a clockwise orientation. As will be shown, this clockwise orientation for regions enables the clockwise order of paths around the point O in the spray γ to be determined efficiently.

Example 3.1 (cont.)

The regions of Figure 3.10 are now given as:

$$\rho_1 = \{1,3,6,5,2\}$$

$$\rho_2 = \{3,4,6\}$$

$$\rho_3 = \{3,1,4\}$$

$$\rho_4 = \{4,2,5,6\}$$

$$\rho_5 = \{2,4,1\}$$

A *search route* for ρ_y is a list of regions that the algorithm has crossed in order to get from the point O to the interior of ρ_y. Each search route obviously begins in a boundary region. If a region has more than one access arc associated with it, a new search route is formed.

Example 3.1 (cont.)

To illustrate this idea the search route, Σ_i for $1 \leq i \leq 3$, of the picture given in Figure 3.10 are:

$$\begin{aligned} \Sigma_1 &= \{\rho_5\} \\ \Sigma_2 &= \{\rho_3, \rho_2\} \\ \Sigma_3 &= \{\rho_2, \rho_4\} \end{aligned}$$

The search route for ρ_3 is therefore given by Σ_2.

The access arcs of the regions encountered by a search route are removed so that a single region, ρ, is created. All the regions whose words were formed from a search route are then contained ρ. The first disc of ρ is the first disc of the arc that bounds the exterior region and the boundary region where the search route began. Note, again, that ρ also has an clockwise orientation.

Example 3.1 (cont.)

The region that merging the search route Σ_2 creates is given in Figure 3.11.

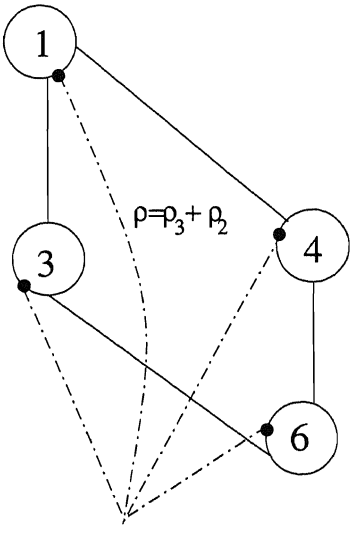


Figure 3.11: The region, ρ, created by merging the regions given by Σ_2

The order of paths in such a region ρ can be determined from the following lemma:

Proposition 3.3.1 *Let ρ be a region as defined above and γ be a spray of a picture* **P**. *The order that the discs whose basepoints lie in ρ are encountered whilst traversing the boundary of ρ in an anti-clockwise direction, gives the order of those paths around γ.*

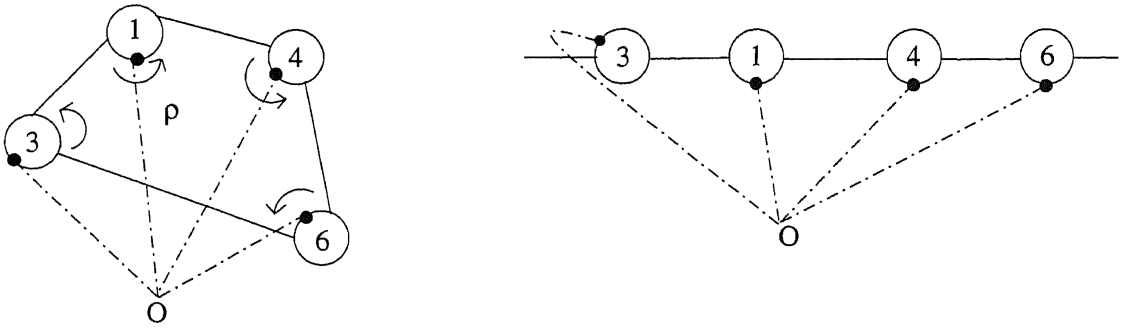


Figure 3.12: Redrawing region ρ

Proof The proof of this proposition is trivial. The access arc into a region ρ can be cut and ρ can be redrawn in a line, as shown in Figure 3.12. This first diagram makes it clear that, because the corners of discs in ρ are traversed in an anti-clockwise direction ρ has a clockwise orientation. This orientation direction gives the order of these paths around γ in a clockwise direction. □

The order of the paths of γ which terminate at discs whose basepoints lie in the external region is found by considering these discs when looking at the boundary regions that contain them.

Example 3.1 (cont.)

The path, that terminates at disc 3 of Figure 3.10 would be examined when looking at the merged regions whose corresponding search routes begin in the boundary regions ρ_2 and ρ_3 (although note that in this example no search routes begin in ρ_3). As can be seen from Figure 3.12, the order of the paths that terminate in the merged region $\rho_2 + \rho_3$ (created from Σ_2) is given by the discs $\{3, 1, 4, 6\}$.

For convenience, the paths are now referred to by the discs they terminate at. The next stage of this algorithm involves combining the paths associated with each boundary region

in such a way that the order of all paths of γ around O that enter the picture through this boundary region are obtained. Assume that there are N search routes associated with the boundary region ρ_x say. The order in which these search routes are given concur with a clockwise traversal around ρ_x. In other words, the search route(s) that leave through the second arc of ρ_x are stored before the search route(s) that leave through the third arc. If more than one search route leaves from the same arc of ρ_x, the order of these routes are also given with a clockwise orientation of the region which they leave to enter different regions. The search routes are put into this order by examining the order of the access arcs associated with each region.

Example 3.1 (cont.)

The search route Σ_1 would therefore be examined before that of Σ_2 in the example depicted in Figure 3.10.

Let κ_i $(1 \leq i \leq N)$ denote the set of discs corresponding to the order of paths around γ that terminate in the region created by merging the regions of search route i. If more than one search route crosses through a particular region then the paths that terminate in this region are examined more than once.

Example 3.1 (cont.)

The paths in Figure 3.10 that terminate at discs 3 and 6 are analysed when looking at the merging of regions $\rho_2 + \rho_4$ and $\rho_3 + \rho_2$.

It is therefore necessary to find a way by which to combine all such sets $\Upsilon_x = \kappa_1 \circ \kappa_2 \circ \cdots \kappa_N$ such that Υ_x gives the correct order of discs whose paths enter the picture through boundary region ρ_x. This method is given by the following lemma:

Lemma 3.3.2 *Let $\Upsilon_x = \kappa_1 \circ \cdots \circ \kappa_N$ be as defined above. Let Δ denote the collection of discs that occur more than once in κ_1 and κ_2. The discs in κ_1 are combined with κ_2 by ensuring that the position of discs with respect to Δ_j $(\forall\ \Delta_j \in \Delta)$ in κ_1 and κ_2 remains constant in $\kappa_1 \circ \kappa_2$.*

In other words, if there is a subset of discs from κ_1 that occurs before Δ_j, say, then this set

of discs occurs before Δ_j in $\kappa_1 \circ \kappa_2$. If both κ_1 and κ_2 have discs occurring before Δ_j, then in $\kappa_1 \circ \kappa_2$ the discs in κ_1 that occur before Δ_j occurs before the subset of discs that occur before Δ_j in κ_2. The same argument can be applied to discs that occur after Δ_j in $\kappa_{1,2}$.

This process is then repeated for $(\kappa_1 \circ \kappa_2) \circ \kappa_3$ until

$$\Upsilon_x = ((\kappa_1 \circ \kappa_2) \circ \kappa_3) \circ \cdots \circ \kappa_N,$$

the set that gives the correct ordering of discs which correspond to the paths around O that enter the picture through boundary region ρ_x, is obtained.

Proof The essence of this proof comes from the fact that the search routes associated with each boundary region are stored with a clockwise orientation. Let Δ_s be a disc that occurs in both κ_i and κ_j, where $j = i + 1$. Assume that the collection of discs, η_i and η_j, occur after Δ_s in κ_i and κ_j respectively, i.e.

$$\begin{aligned} \kappa_i &= \{\cdots, \Delta_s, \eta_i, \cdots\} \\ \kappa_j &= \{\cdots, \Delta_s, \eta_j, \cdots\}. \end{aligned}$$

If $\kappa_i \circ \kappa_j = \{..., \Delta_s, \eta_j, \eta_i, ...\}$ this would imply that the discs in η_j occur before those of η_i around O. The search route giving the order of the discs in κ_i, however, was examined before that of κ_j. As the search routes of each boundary region are constructed in a clockwise order around that boundary region, this method of combining κ_i and κ_j gives an incorrect ordering of discs around O. Similarly, if Δ_s occurs before η_i in κ_i it must occur before it in $\kappa_i \circ \kappa_j$.

The only way in which κ_i can be combined with κ_j to give the correct ordering of discs around O is therefore $\kappa_i \circ \kappa_j = \{, ..., \Delta_s, \eta_i, \eta_j, ..., \}$. The same argument can be applied to discs that occur before Δ_s in κ_i and κ_j. By applying this method to all discs that occur in both κ_i and κ_j the correct ordering around O for the discs in κ_i and κ_j is obtained. Recursively applying this method gives Υ_x. □

This method is applied to boundary region ρ_x and a set Υ_x produced.

Example 3.1 (cont.)

Let $\rho_x = \rho_2$ in Figure 3.10. Here $\kappa_1 = \{3,1,4,6\}$ and $\kappa_2 = \{3,5,6\}$. Discs 3 and 6 occur in both κ_1 and κ_2. Discs 1 and 4 must occur before disc 5 in Υ_2, as the paths given by κ_1 occur (clockwise) before those of κ_2 around O. Similarly, the set $\{1,4,5\}$ must occur after disc 3 and before disc 6. This gives $\Upsilon_2 = \{3,1,4,5,6\}$.

Lemma 3.3.2 is then applied to the next boundary region adjacent anti-clockwise, ρ_{x+1} say, (to adhere to a clockwise ordering around O) to ρ_x and Υ_{x+1} is produced. This procedure continues until all boundary regions have been examined.

Given that there are X boundary regions, there are now X sets of discs:

$$\Upsilon = \Upsilon_x, \Upsilon_{x+1}, \cdots, \Upsilon_{x+X-1}$$

which give the order of paths around O that enter $\mathbf{P}$ through each boundary region. As these boundary regions here examined in an anti-clockwise order, Lemma 3.3.2 is again applied to $((\Upsilon_1 \circ \Upsilon_2) \circ \cdots) \circ \Upsilon_X$ and a list, `IdentitySequence`, is produced. The order of discs in `IdentitySequence` corresponds to the clockwise order of the paths of γ around O that terminate at these discs.

Example 3.1 (cont.)

Applying the above method to the paths in Figure 3.10 gives the following results:

$$\begin{aligned} \Upsilon_5 &= \{2\} \\ \Upsilon_2 &= \{3,1,4,5,6\} \end{aligned}$$

As Υ_5 and Υ_2 are mutually exclusive sets, then `IdentitySequence` $= \{2,3,1,4,5,6\}$. This list then provides the correct ordering of the paths around O in Figure 3.10.

All the information required to produce an identity sequence of a picture is now known: `DiscDetails` gives a list which gives each disc's relator, orientation and region where the

basepoint lies; the word associated with each region has been determined, as has the clockwise order of paths around O. The first disc, Δ_1 say, of `IdentitySequence` is replaced with a list of length two. The first component of this list is the relator that disc Δ_1 represents, R^ϵ say, where $\epsilon = \pm 1$ depending on the orientation of Δ_1. The word associated with the region where the basepoint of Δ_1 lies is stored in the second component. This procedure is repeated for all discs in `IdentitySequence` and, in this way, an identity sequence associated with the picture $\mathbf{P}$ is produced.

The stages of the algorithm can be summarised as follows:

1. Initialise a list `IdentitySequence` that will store the clockwise order of the paths of γ around O.

2. Compute the list `DiscDetails` that gives, for each disc, the region in which its basepoint lies, together with the orientation and relator for that disc.

3. Find the word associated with a region in $\mathbf{P}$. This gives the label of the path from the point O to an interior point of that region. Repeat this process for all regions of $\mathbf{P}$.

4. Create the list `AccessArcs`, where the ith component of `AccessArcs` stores all access arcs from region ρ_i.

5. Create search routes that store the regions which are encountered when traversing each path from O to an interior point of a region. Select a boundary region, ρ_x.

   (a) For every search route from a boundary region ρ_x, create a 'merged region' by removing the arcs used to access the regions encountered by the search route. This merged region is traversed in an anti-clockwise direction, to give a clockwise orientation of the region, and the discs who have a basepoint in this region, or the exterior region, are stored in the order that they are encountered.

   (b) Combine these discs from all search routes associated with ρ_x according to the criteria given in Lemma 3.3.2

6. Select the next adjacent, anti-clockwise, boundary region. Repeat steps (a) and (b) of item (5) for all search routes associated with this boundary region.

7. Repeat (6) for all boundary regions.

8. Combine the paths of all boundary regions according to the criteria given in Lemma 3.3.2 and store the order of discs around O in a list `IdentitySequence`

9. Use the information in `IdentitySequence` and `DiscDetails` to compute the identity sequence associated with **P**.

3.4 Implementation, Experiments and Conclusions

The GAP algorithm `PictureFromIdentitySequence` takes as input a finitely presented group, $\mathcal{P}$, together with a list of identity sequences, $\mathcal{L}$, of $\mathcal{P}$ and returns a list, `RelatorInformation` and a rotation scheme for each picture in $\mathcal{L}$. The `IdentitySequenceFromPicture` function takes a presentation and a representation of a picture over this presentation, given by `RotationScheme` and `RelatorInformation`, and produces an identity sequence that this picture depicts.

The sets of identity sequences for the 45 non-abelian groups of order less than 32, have been presented by Ellis and Kholodna [24]. Each of these 45 sets sequences were used to test the `PictureFromIdentitySequence` program. Five of these sets, however, were returned as invalid identity sequences by the program. Further investigations, however, showed that this was not due to an error in the code but due to typographical errors given in the paper [24]. Once the errors were rectified, the code produced picture representations that agreed with manually produced results. For each representation produced from `PictureFromIdentitySequence`, the function `IdentitySequenceFromPicture` was called. In all cases the sequences produced equalled the identity of the free group underlying the given presentation.

Example 3.2

To illustrate these results, an example is given for a presentation of the group Q_4, $\mathcal{P} =< a, b; a^{-2}b^4, a^{-1}bab >$. Ellis and Kholodna showed that this presentation has only one identity sequence that generates π_2. Given that $R = a^{-2}b^4$ and $S = a^{-1}bab$, this

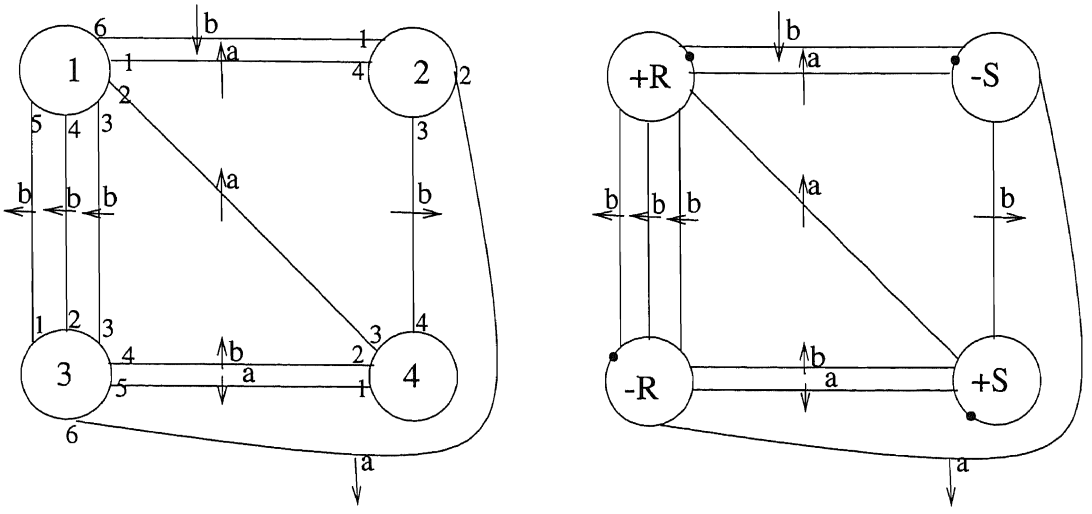


Figure 3.13: A visualisation of the rotation scheme for Q_4

sequence is:

$$(aRa^{-1}, aS^{-1}a^{-1}, baR^{-1}a^{-1}b^{-1}, bSb^{-1})$$

This sequence was given, in the required form, as input for the program `PictureFromIdentitySequence`. The lists `RelatorInformation` and `RotationScheme` were produced for the discs in this picture. Given that `f1` $= a$ and `f2` $= b$, the GAP output for the rotation scheme of the picture is as follows:

```
[ [ [ 2, 4, f1^-1 ], [ 4, 3, f1^-1 ], [ 3, 3, f2 ], [ 3, 2, f2 ],
    [ 3, 1, f2 ], [ 2, 1, f2 ] ],
  [ [ 1, 6, f2^-1 ], [ 3, 6, f1^-1 ], [ 4, 4, f2^-1 ], [ 1, 1, f1 ] ],
  [ [ 1, 5, f2^-1 ], [ 1, 4, f2^-1 ], [ 1, 3, f2^-1 ], [ 4, 2, f2^-1 ],
    [ 4, 1, f1 ], [ 2, 2, f1 ] ],
  [ [ 3, 5, f1^-1 ], [ 3, 4, f2 ], [ 1, 2, f1 ], [ 2, 3, f2 ] ] ]
```

Similarly `RelatorInformation` is given as :

```
[rec(Relator := f1^-2*f2^4, PositiveDiscs := [ 1 ], NegativeDiscs := [ 3 ]),
rec(Relator := f1^-1*f2*f1*f2, PositiveDiscs := [ 4 ], NegativeDiscs := [ 2 ])]
```

A visualisation of this information is given by the diagrams in Figure 3.13. The first picture

shows the realisation of the rotation scheme and `RelatorInformation`. The more standard picture over Q_4 is given in the second diagram of Figure 3.13, where labels have been chosen for discs and the location of their basepoints selected.

The function `IdentitySequenceFromPicture` took the above list `RelatorInformation` and the rotation scheme as input and produced the following sequences:

```
IdSequence:= [ [ f2^-4*f1^2, <identity ...> ], [ f1^-1*f2*f1*f2, f1^-1 ],
              [ f2^-1*f1^-1*f2^-1*f1, f2 ], [ f1^-2*f2^4, f2 ] ]
```

Here `<identity ...>` denotes the identity of the free group underlying Q_4. These sequences are indeed identity sequences as their product equals the identity of F as the following GAP routine shows:

```
gap> for i in [1..Length(IdSequence)] do
> IdSequence[i]:=IdSequence[i][2]*IdSequence[i][1]*IdSequence[i][2]^-1;
> od;
gap> IdSequence;
[ f2^-4*f1^2, f1^-2*f2*f1*f2*f1, f1^-1*f2^-1*f1*f2^-1, f2*f1^-2*f2^3 ]
gap> Product(IdSequence);
<identity ...>
```

Note, however, that this sequence is not identically equal to the original sequence provided by Ellis and Kholodna. This is due to the fact that this program has chosen a different exterior region. To produce the same sequence the region that contains the discs $+R, +S$, and $-S$, as shown in the second diagram of Figure 3.13, should be chosen as the exterior region.

These programs provide a useful way to obtain further information about the structure of π_2. Given a presentation, a set of identity sequences can be computed using the packages given in [24] and [34]. `PictureFromIdentitySequence` can then be used to gain a representation of the corresponding pictures. Furthermore `PictureFromIdentitySequence` also provides a method to verify that a given sequence is indeed a 'consequence of the relators'; if a rotation

scheme cannot be determined from the sequence then the sequence is not a element of π_2. Similarly, `IdentitySequenceFromPicture` provides a easy way to determine the identity sequence from a representation of a picture. These identity sequences can then be used to gain further information about the group presentation they represent. One of the main advantages of these programs is that, given an algorithm that requires π_2 elements as input, this data can now be given either geometrically or algebraically.

Moreover, the work described in this chapter could be used to automatically generate π_2 generators of a presentation. There are automatic graph drawings programs, such as the those used in [17] to test algorithms upon and the GDToolkit [32], which is a software package that automatically produces graphs according to a desired aesthetic criteria. The methods that these programs adopt could be adapted, and combined with the work in Chapters 5 and 6, produce representations for pictures over a presentation, $\mathcal{P}$ say. For instance, a certain number of each discs for each relator in $\mathcal{P}$ could be specified and a program could then attempt to produce a picture over $\mathcal{P}$. The `IdentitySequenceFromPicture` function could then be utilised to determine if this picture represents an element in the second homotopy module of $\mathcal{P}$. Further algebraic techniques could be introduced to investigate if this element was a generator of π_2. Ultimately it is hoped that a user could enter a presentation into a program and a number of π_2 generators for this presentation would then be computed. The final stage in this plan is that this list of generators would become a minimal generating set for π_2.

Chapter 4

Group Extensions

The focus of this chapter is to present a geometric algorithm. This technique implements the method given in [1] which uses the geometry of pictures to determine if a given presentation does define a genuine group extension.

4.1 Group Extensions

Definition 4.1 (Group Extension) *Let K and Q be groups. A group, H, is said to be an extension of K by Q if there is a short exact sequence*

$$1 \to K \to H \to Q \to 1.$$

Let $\mathcal{Q} =< \mathbf{a}; \mathbf{r} >$ and $\mathcal{K} =< \mathbf{x}; \mathbf{v} >$ be presentations for the non-trivial groups Q and K respectively. Any extension, H, of K by Q will have a presentation of the form:

$$\mathcal{H} =< \mathbf{a}, \mathbf{x}; RW_R^{-1}(R \in \mathbf{r}), \mathbf{v}, a^{-\epsilon}xa^{\epsilon}(x^{a^{\epsilon}})^{-1}(x \in \mathbf{x}, a \in \mathbf{a}, \epsilon = \pm 1) > \tag{4.1}$$

(see [44] for further information). Here, W_R $(R \in \mathbf{r})$ and $x^{a^{\epsilon}}$ $(x \in \mathbf{x}, a \in \mathbf{a}, \epsilon = \pm 1)$ are non-empty reduced words on $\mathbf{x}$. The words $a^{-\epsilon}xa^{\epsilon}(x^{a^{\epsilon}})^{-1}$ with $\epsilon = +1$ are known as the *positive endomorphisms* of $\mathcal{H}$, and those with $\epsilon = -1$ are the *negative endomorphisms*. In

general a presentation of the form (4.1) gives an exact sequence:

$$G(\mathcal{K}) \xrightarrow{i} G(\mathcal{H}) \xrightarrow{p} G(\mathcal{Q}) \to 1 \tag{4.2}$$

where i is induced by the mapping $x \mapsto x$ ($x \in \mathbf{x}$), and p is induced by $a \mapsto a$, $x \mapsto 1$ ($a \in \mathbf{a}, x \in \mathbf{x}$). In order for (4.1) to define an extension of K by Q, it is required that i is injective. If this is the case then $\mathcal{H}$ is said to be *admissible.* When $\mathcal{H}$ is admissible it can be shown that the negative endomorphisms are superfluous, that is, if

$$\hat{\mathcal{H}} = < \mathbf{a}, \mathbf{x}; RW_R^{-1}(R \in \mathbf{r}), \mathbf{v}, a^{-1}xa(x^a)^{-1}(x \in \mathbf{x}, a \in \mathbf{a}) >$$

then $G(\hat{\mathcal{H}}) = G(\mathcal{H})$. The presentation $\hat{\mathcal{H}}$ is called the *reduced presentation.* These definitions for $\mathcal{H}$, $\hat{\mathcal{H}}$, $\mathcal{Q}$ and $\mathcal{K}$ will remain fixed throughout this chapter.

4.2 Test Words

Let $Pict_{\mathbf{x}}(\mathcal{H})$ denote the set of pictures over $\mathcal{H}$ with boundary label a word on $\mathbf{x}$. It can be shown, (see [1] for example), that $\mathcal{H}$ is admissible if and only if for each picture $\mathbf{P} \in Pict_{\mathbf{x}}(\mathcal{H})$ there is a picture *over* $\mathcal{Q}$ with the same boundary label. It is shown further that to check admissibility, it is only necessary to consider certain special elements of $Pict_{\mathbf{x}}(\mathcal{H})$, which are described in [1]. These special pictures fall into four types. The boundary labels of these pictures are called the *test words*, $\mathbf{t}(\mathcal{H})$, for admissibility. Thus $\mathbf{t}(\mathcal{H})$ is the union of four sets.

Before these sets can be described some notation is required. For any word $W = x_1^{\epsilon_1} x_2^{\epsilon_2} \cdots x_n^{\epsilon_n}$ ($x_i \in \mathbf{x}, \epsilon_i = \pm 1$, $1 \leq i \leq n$) on $\mathbf{x}$ and for $a \in \mathbf{a}, \epsilon = \pm 1$, the word W^{a^ϵ} is defined as:

$$W^{a^\epsilon} = x_1^{a^{\epsilon_1}} x_2^{a^{\epsilon_2}} \cdots x_n^{a^{\epsilon_n}}.$$

Then if $U = a_1^{\delta_1} a_2^{\delta_2} \cdots a_m^{\delta_m}$ ($a_i \in \mathbf{a}, \delta_i = \pm 1$, $1 \leq i \leq m$) is a word on $\mathbf{a}$ of length greater than 1, one writes $U = V a_m^{\delta_m}$, and W^U is inductively defined to be $(W^V)^{a_m^{\delta_m}}$. The word

W^{-U} means $(W^U)^{-1}$. The four sets of test words can now be described:

1. **Test words of type 1:** $\mathbf{t}_1^\epsilon(\mathcal{H}) = \{x^{-1}x^{a^\epsilon a^{-\epsilon}} : x \in \mathbf{x}, a \in \mathbf{a}\}$ $(\epsilon = \pm 1)$.

2. **Test words of type 2:** $\mathbf{t}_2(\mathcal{H}) = \{xW_R x^{-R} W_R^{-1} : x \in \mathbf{x}, R \in \mathbf{r}\}$

3. **Test words of type 3:** Let $< \mathcal{Q}; \mathbf{s} >$ be a *3-presentation* for Q. For the purposes of this work it is assumed that $\mathbf{s}$ is given algebraically (i.e. $\mathbf{s}$ is a generating set of identity sequences), although $\mathbf{s}$ could just as easily be given in a geometric form. For $s \in \mathbf{s}$ one has

$$s = (U_1 R_1^{\epsilon_1} U_1^{-1}, U_2 R_2^{\epsilon_2} U_2^{-1}, \cdots, U_m R_m^{\epsilon_m} U_m^{-1})$$

where U_i is a word on $\mathbf{a}$, $R_i \in \mathbf{r}$ and $\epsilon_i = \pm 1$ for $1 \leq i \leq m$. One lets $W(s) = W_{R_m}^{\epsilon_m U_m} ... W_{R_1}^{\epsilon_1 U_1}$. Then $\mathbf{t}_3(\mathcal{H}) = \{W(s) : s \in \mathbf{s}\}$.

4. **Test words of type 4:** $\mathbf{t}_4^\epsilon(\mathcal{H}) = \{V^{a^\epsilon} : a \in \mathbf{a}, V \in \mathbf{v}\}$ $(\epsilon = \pm 1)$.

Note that the first three sets of test words do not require the presentation of K. These test words can all be determined from the "partial presentation":

$$\mathcal{H}^\circ = < \mathbf{a}, \mathbf{x}; RW_R^{-1} (R \in \mathbf{r}), a^{-\epsilon} x a^\epsilon (x^{a^\epsilon})^{-1} (x \in \mathbf{x}, a \in \mathbf{a}, \epsilon = \pm 1) >$$

together with a 3-presentation for $\mathcal{Q}$.

It transpires that $\mathcal{H}$ is admissible if and only if each element of the set $\mathbf{t}(\mathcal{H})$ defines the identity of K (as proven in [1], Theorem 6.1).

4.3 Algorithm for Admissibility

Let $\mathbf{x}$ be an alphabet, K be a group and let

$$\mu : \mathbf{x} \to K \qquad x \mapsto k_x,$$

be a function. It is required that the elements k_x $(x \in \mathbf{x})$ generate K. For a non-empty word

$$W = x_1^{\epsilon_1} x_2^{\epsilon_2} \cdots x_n^{\epsilon_n} \quad (x_i \in \mathbf{x}, \epsilon_i = \pm 1, 1 \leq i \leq n)$$

on $\mathbf{x}$, $\mu(W)$ is defined to be the product $k_1^{\epsilon_1} k_2^{\epsilon_2} \cdots k_n^{\epsilon_n}$ in K. If W is the empty word then $\mu(W)$ is defined to be the identity of K. It is assumed that a *3-presentation*, $< \mathbf{a}; \mathbf{r}; \mathbf{s} >$, for a group Q is given. The method to determine if a presentation, $\mathcal{H}$, defines an extension of K by Q is now described.

The algorithm begins by designating a "partial presentation",

$$\mathcal{H}^\circ = < \mathbf{a}, \mathbf{x}; RW_R^{-1} (R \in \mathbf{r}), a^{-\epsilon} x a^{\epsilon} (x^{a^{\epsilon}})^{-1} (x \in \mathbf{x}, a \in \mathbf{a}, \epsilon = \pm 1) >$$

which is defined by specifying a list of words W_R $(R \in \mathbf{r})$ and lists of positive and negative endomorphisms. The algorithm then proceeds as follows:

Step 1: For each $a \in \mathbf{a}$ compute the set of test words

$$\mathbf{t}_1^{+1}(\mathcal{H}^\circ) = \{x^{-1} x^{aa^{-1}} : x \in \mathbf{x}, a \in \mathbf{a}\}.$$

For each word, $T \in \mathbf{t}_1^{+1}(\mathcal{H}^\circ)$, compute $\mu(T)$. If each $\mu(T) = 1$ then proceed to Step 2. If some $\mu(T) \neq 1$ then stop.

Step 2: For each $a \in \mathbf{a}$ compute the set of test words

$$\mathbf{t}_1^{-1}(\mathcal{H}^\circ) = \{x^{-1} x^{a^{-1}a} : x \in \mathbf{x}, a \in \mathbf{a}\}.$$

For each word, $T \in \mathbf{t}_1^{-1}(\mathcal{H}^\circ)$, compute $\mu(T)$. If each $\mu(T) = 1$ then proceed to Step 3. If some $\mu(T) \neq 1$ then stop.

Step 3: For each $x \in \mathbf{x}$ and $R \in \mathbf{r}$ compute the set of test words

$$\mathbf{t}_2(\mathcal{H}^\circ) = \{xW_R x^{-R} W_R^{-1} : x \in \mathbf{x}, R \in \mathbf{r}\}.$$

For each word, $T \in \mathbf{t}_2(\mathcal{H}^\circ)$, compute $\mu(T)$. If each $\mu(T) = 1$ then proceed to Step 4. If some $\mu(T) \neq 1$ then stop.

Step 4: For each $s \in \mathbf{s}$ compute the set of test words

$$\mathbf{t}_3(\mathcal{H}^\circ) = \{W(s) : s \in \mathbf{s}\}.$$

For each word, $T \in \mathbf{t}_3(\mathcal{H}^\circ)$, compute $\mu(T)$. If each $\mu(T) = 1$ then proceed to the next stage. If some $\mu(T) \neq 1$ then stop.

If all four of the above steps are successful then $\mathcal{H}^\circ$ is a valid "partial presentation". The presentation is "completed" by introducing a presentation $\mathcal{K} =< \mathbf{x}; \mathbf{v} >$ for K corresponding to the generators $\{k_x : x \in \mathbf{x}\}$. This then gives the presentation

$$\mathcal{H} =< \mathbf{a}, \mathbf{x}; RW_R^{-1}(R \in \mathbf{r}), \mathbf{v}, a^{-\epsilon} x a^{\epsilon} (x^{a^{\epsilon}})^{-1} (x \in \mathbf{x}, a \in \mathbf{a}, \epsilon = \pm 1) > .$$

Step 5: For each $a \in \mathbf{a}$ and $V \in \mathbf{v}$ compute the set of test words

$$\mathbf{t}_4^{+1}(\mathcal{H}) = \{V^a : a \in \mathbf{a}, V \in \mathbf{v}\}.$$

For each word, $T \in \mathbf{t}_4^{+1}(\mathcal{H})$, compute $\mu(T)$. If each $\mu(T) = 1$ then proceed to Step 6. If some $\mu(T) \neq 1$ then stop.

Step 6: For each $a \in \mathbf{a}$ and $V \in \mathbf{v}$ compute the set of test words

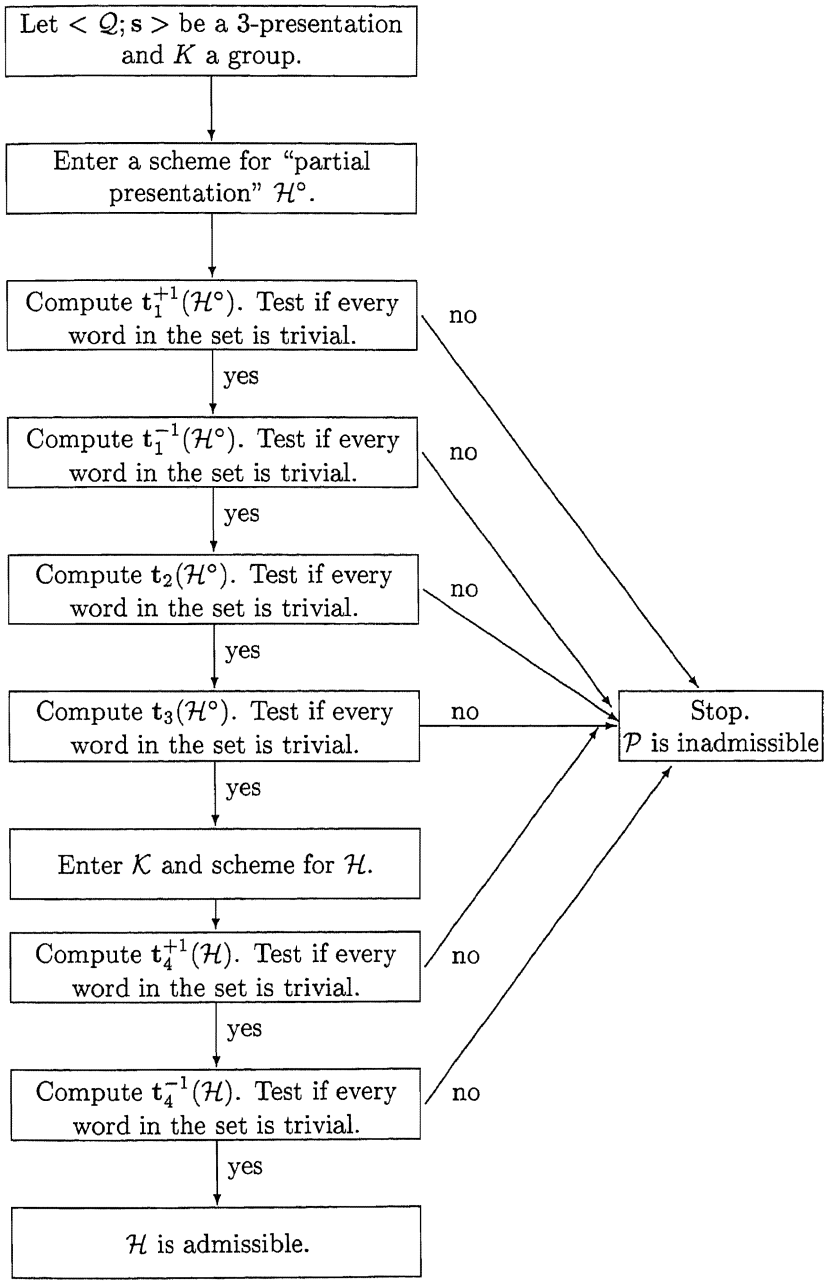
$$\mathbf{t}_4^{-1}(\mathcal{H}) = \{V^{a^{-1}} : a \in \mathbf{a}, V \in \mathbf{v}\}.$$

For each word, $T \in \mathbf{t}_4^{-1}(\mathcal{H})$, compute $\mu(T)$. If each $\mu(T) = 1$ then $\mathcal{H}$ is admissible. If some $\mu(T) \neq 1$ then stop.

If these final two steps are successful then $\mathcal{H}$ is admissible.

The algorithm to determine if a given scheme of a group presentation is admissible is defined schematically in Figure 4.1.

Figure 4.1: A diagram of the algorithm to determine admissibility



A 3-presentation for H can be obtained by implementing the results given and proved in Theorem 6.4 of [1]:

Theorem 4.3.1 *(Baik et al.) If Q and K have finite 3-presentations $< \mathcal{Q}; \mathbf{s} >$ and $< \mathcal{K}; \mathbf{y} >$ respectively and $\mathcal{H}$ is admissible then the group H defined by $\mathcal{H}$ has 3-presentation*

$$< \hat{\mathcal{H}}; \mathbf{y}, S(\mathbf{B}_{V,a}), S(\mathbf{D'}_{x,R}), S(s') >$$

where $a \in \mathbf{a}, x \in \mathbf{x}, V \in \mathbf{v}, R \in \mathbf{r}$ and $s' \in \mathbf{s}$.

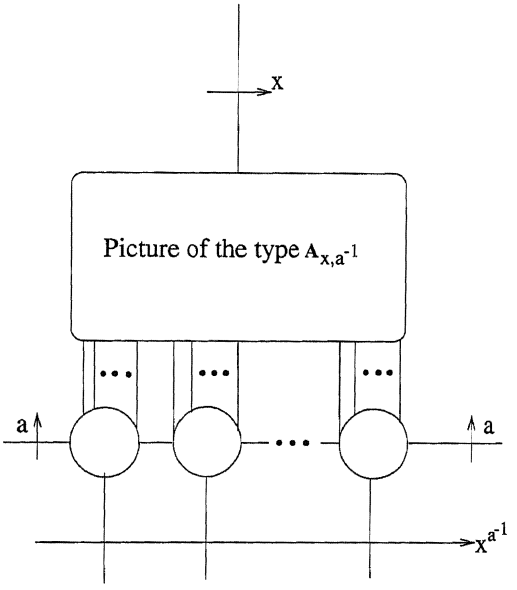


Figure 4.2: The subpicture replacing discs labelled by the negative endomorphisms

Here the spherical pictures $S(\mathbf{B}_{V,a})$ are obtained by the disjoint union of the picture $\mathbf{B}_{V,a}$ together with its mirror image, $-\mathbf{B}_{V,a}$. This reasoning also lies behind the pictures $S(\mathbf{D'}_{x,R})$ and $S(s')$, where $\mathbf{D'}_{x,R}$ and s' are specific types of pictures over $\mathcal{H}'$. The pictures $\mathbf{D'}_{x,R}$ and s' are produced from $\mathbf{D}_{x,R}$ and s' respectively, by replacing each disc labelled with the negative endomorphisms by the picture in Figure 4.2. The number of subpictures to be added to this picture becomes exponential in growth: if there are m discs corresponding to the negative endomorphisms in $\mathbf{D}_{x,R}$ and $x^{a^{-1}}$ has length N then there are $\sum_{i=0}^{(m-1)} N^i$ subpictures, of the form given in Figure 4.2, to be added to $\mathbf{D}_{x,R}$ to produce $\mathbf{D'}_{x,R}$. As can be imagined, even in the simplest of cases, the pictures $\mathbf{D'}_{x,R}$ and s' can be extremely complicated. This also applies to the formulae that would produce the equivalent identity

sequences. To implement the results from the above theorem is a project in itself and as such is left for further work. For this reason it was decided to abandon the attempt to implement the results in the above theorem. It should be observed that a technique given in [24] computes 3-presentations of abelian group extensions; the result cited here would be useful to implement, nevertheless, as it provides a method to compute 3-presentations of general group extensions.

4.4 Implementation, Experiments and Conclusions

An algorithm is presented, based upon the above method, that determines if a given presentation, $\mathcal{H}$, defines an extension of the group K by Q. The algorithm takes as input the 3-presentation $< Q; \mathbf{s} >$ for a group Q, either a group K or a 2-presentation for K, the mapping μ and and a scheme for the "partial presentation", $\mathcal{H}^\circ$, of $\mathcal{H}$. This scheme consists of the positive and negative endomorphisms and the words of the form W_R. The set $\mathbf{s}$ can either be given as a set of identity sequences or spherical pictures. If the latter form is used, the pictures can be inputted by means of their data structures, as described in Chapter 2, or drawn as graphic objects using the SPICE editor (see Chapter 6 for more information).

In order to enter 3-presentations, a GAP object `3Presentation` was created. This takes the 2-presentation together with either a representation of a set of pictures, or a set of identity sequences. In the first case the `IdentitySequencesFromPictures` function, as described in Chapter 3, can be used to provide the corresponding identity sequences from a set of spherical pictures. Note that the packages of Ellis and Kholodna [24] or Heyworth and Wensley [34] could be used to provide a generating set of identity sequences for Q. The identity sequences must be of the form as described in Chapter 3, that is the term $U_i R_i^{\epsilon_i} U_i^{-1}$ is given by $[R_i^{\epsilon_i}, U_i]$.

The option of entering K is given as it is generally easier to input a group rather than a group presentation, $\mathcal{K}$, as no calculations are needed. The information given in $\mathcal{K}$ is only required for the fourth set of test words (Steps 5 and 6). If a given scheme fails on Steps 1-4 there is no need for $\mathcal{K}$. For convenience, it is assumed that K is given initially.

It is required that $\mathcal{H}^\circ$ must be given in such a way that the map μ, the action, (the positive and negative endomorphisms) and the set $W_{\mathbf{r}}$ are specified. This is achieved by entering all the data required for this algorithm by means of a record. The record has the following components `3Presentation`, `Group` (or `GroupPresentation` if $\mathcal{K}$ is given), `Map`, `PositiveEndomorphism`, `NegativeEndomorphism` and `WR`. It is remarked that while it is not standard practice to consider the negative endomorphisms of group extensions, they are obviously required here to ensure that all possible test words are computed.

Lists are used to input both the map μ and the scheme for $\mathcal{H}^\circ$. Recall that $\mu : \mathbf{x} \to K$, $x \mapsto k_x$. The list `Map` contains the range of μ whereby k_{x_i} is stored at the ith position of this list ($1 \leq i \leq |\mathbf{x}|$). The set $W_{\mathbf{r}}$ is given as a list where the word W_{R_i} for $R_i \in \mathbf{r}$ is located at the ith position.

The positive endomorphisms are entered by means of a list of lists. For every generator of $\mathcal{Q}$, there is a corresponding list in `PositiveEndomorphism`. The jth list of `PositiveEndomorphism` contains the words $(x^{a_j})^{-1}$ for every $x \in \mathbf{x}, 1 \leq j \leq |\mathbf{a}|$ i.e. the word $(x_i^{a_j})^{-1}$ is stored at the ith position of this jth list. The list for the negative endomorphisms has a similar structure.

Example 4.1

Let Q be the cyclic group of order 3. A 3-presentation for Q is given by $< a; a^3; (R, a^{-1}R^{-1}a) >$, where R denotes the relator a^3. Let $\mathbf{x} = \{x, y\}$ and K be the cyclic group of order 14 generated by k, and let $\mu : \mathbf{x} \to K$ be given by $x \mapsto k^7$ and $y \mapsto k$. A possible "partial presentation" is as follows:

$$\mathcal{H}^\circ =< a, x, y; a^3y^{-7}, a^{-1}xax^{-1}y^{-1}x^{-1}yx^{-1}, axa^{-1}x, a^{-1}yay^{-4}x^{-1}, aya^{-1}y^{-1}x^{-1}y^{-1} > .$$

The following lists would be used to input μ and the structure of $\mathcal{H}^\circ$:

$$\begin{aligned}
[\ & \texttt{Map} := [k^7, k] \\
& \texttt{PositiveEndomorphisms} := [[xy^{-1}xyx, xy^4]] \\
& \texttt{NegativeEndomorphisms} := [[x^{-1}, yxy]]
\end{aligned}$$

$$\texttt{WR} := [y^7] \ ]$$

(This example will be continued later - see Example 4.4.6)

This algorithm, to determine if a given presentation defines a group extension, is implemented in the functions `AdmissiblePresentationWithGrp`, `AdmissiblePresentation` and `AdmissiblePresentationsWithFpGrps`.

`AdmissiblePresentationWithGrp` takes a record containing the scheme for the partial presentation $\mathcal{H}^\circ$, a 3-presentation, $< \mathcal{Q}; \mathbf{s} >$ and the group K. This function then carries out Steps 1-4, as described above. If the scheme fails on any of these steps, the test word that the scheme failed upon is returned. Returning the test word in this way enables the user to manipulate the scheme to produce an admissible presentation. If $\mathcal{H}^\circ$ is a valid "partial presentation", the free group, $\mathbf{h}$, underlying the "partial presentation" $\mathcal{H}^\circ$ is returned. Returning $\mathbf{h}$ in this operation saves computation time when determining the fourth set of test words.

This last set of words are determined with the function `AdmissiblePresentation`, which takes $\mathbf{h}$, a presentation, $\mathcal{K}$, for K, a scheme for a "complete" presentation $\mathcal{H}$ and $< \mathcal{Q}; \mathbf{s} >$ as input. This presentation is now added into the input for this algorithm by means of the record component `GroupPresentation`. A "complete" presentation $\mathcal{H}$ can now be calculated.

Steps 5 and 6 are now implemented and if these steps are successful the algorithm concludes that $\mathcal{H}$ is indeed a presentation of a group extension and $\mathcal{H}$ is returned in the form of a GAP presentation. The complete presentation $\mathcal{H}$ is returned because the negative endomorphisms of $\mathcal{H}$ are needed if a 3-presentation for H is required. Depending on the requirements of the user, the function `ReducedFpGrpExtension` can be called which returns the reduced presentation, $\hat{\mathcal{H}}$, for H. Note that if Steps 5 and 6 are not successful the test word that the scheme failed upon is again returned.

If a presentation for $\mathcal{K}$ is know initially, the function `AdmissiblePresentationsWithFpGrps` can be used. The procedure takes a record containing the scheme for $\mathcal{H}$, a 3-presentation,

$< Q; s >$, and a 2-presentation, $\mathcal{K}$, for K and determines if $\mathcal{H}$ defines an extension of K by Q. Again, if this is the case the presentation $\mathcal{H}$ is produced; alternatively, the scheme the test word failed upon is returned.

These algorithms have been tested on various examples, the results of which are shown below. The algorithms require that the data is entered in a specific way. It is felt that using a record to enter all the data, however, makes the algorithms as "user-friendly" as possible. Furthermore, there are several options open to the user: if a group presentation for K is not known, the user only needs to enter K for the first four stages of the algorithm. A presentation is only required if these four stages are passed. Similarly, if $\mathcal{H}$ is a presentation of a group extension, a user can choose whether to use this complete presentation or the more standard reduced presentation, $\hat{\mathcal{H}}$.

In order to implement these algorithms efficiently, the GAP object `3Presentation` was created, together with several new functions. These functions enable the user to obtain information about the 3-presentation, such as the corresponding 2-presentation, and the generators, relators and generating set of the 3-presentation. Appendix A explains these functions in more detail. Whilst it is not a complex matter to implement these functions, they are useful for this work and they can be used independently. Indeed they could become a useful tool when investigating 3-presentations.

In conclusion therefore, this chapter presents a algorithm into determining if a given presentation defines a group extension. The usefulness of these functions would be increased if Theorem 6.4 of Baik *et al.* [1] was implemented, which would provide a way to automatically compute 3-presentations of group extensions.

4.4.1 Examples

The GAP algorithm was used to test if certain presentations were valid group extensions. In the following examples Q is the cyclic group of order 3. A 3-presentation for Q is given by $< a; a^3; (R, a^{-1}R^{-1}a) >$, where R denotes the relator a^3. Various choices are made for K. All results agree with manual computations. (These examples are rather contrived, but

were created for test purposes.)

Example 4.4.1

Let K be the subgroup of $GL_2(\mathbf{Q})$ generated by the matrices $k_1 = \begin{bmatrix} 1 & 6 \\ -2 & 4 \end{bmatrix}$ and $k_2 = \begin{bmatrix} 3 & 4 \\ -1 & -2 \end{bmatrix}$. Let $\mathbf{x} = \{x, y\}$ and $\mu : x \mapsto k_1, y \mapsto k_2$.

A "partial presentation" is given by:

$$\mathcal{H}^\circ = < a, x, y; a^3y^{-6}x^{-2}, a^{-1}xax^{-1}y^{-1}x^{-1}yx^{-1}, axa^{-1}x^{-1}, a^{-1}yay^{-1}x^{-1}, aya^{-1}y^{-1}x^{-1}y^{-1} > .$$

This scheme fails at Step 1 on the test word $x^{-1}x^{aa^{-1}}$ and therefore $\mathcal{H}^\circ$ is inadmissible.

Example 4.4.2

Let K be the alternating group A_4 generated by $k_1 = (12)(34)$, $k_2 = (123)$. Let $\mathbf{x} = \{x, y\}$ and $\mu : x \mapsto k_1, y \mapsto k_2$.

A "partial presentation" is given by:

$$\mathcal{H}^\circ = < a, x, y; a^3y^{-3}x, a^{-1}xay^{-1}x^{-1}, axa^{-1}x^{-1}, a^{-1}yax^{-2}y^{-1}, aya^{-1}(xy)^{-3} > .$$

This scheme fails at Step 2 on the test word $x^{-1}x^{a^{-1}a}$ and therefore $\mathcal{H}^\circ$ is inadmissible.

Example 4.4.3

Let K be the subgroup of S_7 generated by the permutations $k_1 = (1234)(567)$ and $k_2 = (12)(34)$. Let $\mathbf{x} = \{x, y\}$ and $\mu : x \mapsto k_1, y \mapsto k_2$.

A "partial presentation" is given by:

$$\mathcal{H}^\circ =< a, x, y; a^3x^{-4}, a^{-1}xay^{-1}x^{-1}y^{-1}, axa^{-1}y^{-1}x^{-1}y^{-1}, a^{-1}yay, aya^{-1}y >.$$

This scheme fails at Step 3 on the test word $x(x^4)x^{-a^3}(x^{-4})$ and therefore $\mathcal{H}^\circ$ is inadmissible.

Example 4.4.4

Let K be the cyclic group of order 14 generated by k_1. Let $\mathbf{x} = \{x, y\}$ and $\mu : x \mapsto k_1^{-7}$, $y \mapsto k_1$.

A "partial presentation" is given by:

$$\mathcal{H}^\circ =< a, x, y; a^3y^{-6}x^{-2}, a^{-1}xax^{-1}y^{-1}x^{-1}yx^{-1}, axa^{-1}x, a^{-1}yay^{-4}x^{-1}, aya^{-1}y^{-1}x^{-1}y^{-1} >.$$

This scheme fails at Step 4 when s equals the identity sequence $a^{-1}Ra$. Therefore, $\mathcal{H}^\circ$ is inadmissible.

Example 4.4.5

Let K be the subgroup of S_7 generated by the permutation $k_1 = (1234)(567)$ and $k_2 = (12)(34)$. Let $\mathbf{x} = \{x, y\}$ and $\mu : x \mapsto k_1, y \mapsto k_2$.

A "partial presentation" is given by:

$$\mathcal{H}^\circ =< a, x, y; a^3y^{-1}x^4, a^{-1}xay^{-1}x^{-1}y^{-1}, axa^{-1}y^{-1}x^{-1}y^{-1}, a^{-1}yay, aya^{-1}y >.$$

This scheme passes on Steps 1-4. A presentation,

$$\mathcal{K} =< x, y; (xy)^6, y^2, x^{-1}yx^2(xy)^5, y^{-1}x^3(xy)^3, x^{-2}yx^2y >$$

for K can be computed using GAP. A complete presentation for $\mathcal{H}$ can now be computed.

This scheme passes on Steps 5 and 6 and therefore

$$\mathcal{H} = < a,x,y; a^3y^{-1}x^4, a^{-1}xay^{-1}x^{-1}y^{-1}, axa^{-1}y^{-1}x^{-1}y^{-1}, a^{-1}yay, aya^{-1}y, (xy)^6, y^2, x^{-1}yx^2(xy)^5, y^{-1}x^3(xy)^3, x^{-2}yx^2y$$

is is a presentation for an extension of K by Q.

Example 4.4.6 (Example 4.1 (cont.)

Let K be the cyclic group of order 14 generated by k_1. Let $\mathbf{x} = \{x,y\}$ and $\mu : x \mapsto k_1^{-7}$, $y \mapsto k_1$.

A "partial presentation" is given by:

$$\mathcal{H}^\circ =< a,x,y; a^3y^{-7}, a^{-1}xax^{-1}y^{-1}x^{-1}yx^{-1}, axa^{-1}x, a^{-1}yay^{-4}x^{-1}, aya^{-1}y^{-1}x^{-1}y^{-1} > .$$

This scheme passes on Steps 1-4. A presentation $\mathcal{K}$ for K is given by $< x,y; y^{14}, xy^7 >$. A complete presentation for $\mathcal{H}$ can now be computed.

This scheme passes on Steps 5-6 and therefore

$$\mathcal{H} =< a,x,y; a^3y^{-7}, a^{-1}xax^{-1}y^{-1}x^{-1}yx^{-1}, axa^{-1}x, a^{-1}yay^{-4}x^{-1}, aya^{-1}y^{-1}x^{-1}y^{-1}, y^{14}, xy^7 >$$

is a presentation for an extension of K by Q.

Chapter 5

The Spherical Picture Editor

The work presented in this chapter describes the features of both picture objects and the Spherical PICture Editor (SPICE). The methods utilised to create and manipulate graphical picture objects are provided together with the capabilities of the editor. SPICE also contains a database of generating pictures for the presentations of groups of order less than 32. The features of this database, and its future potential, are also detailed in this chapter.

SPICE utilises the XGAP share package, which is the existing graphical interface for GAP. An advantage of using XGAP is that it contains many features, such as circles and lines, that can be used to create discs and arcs. Pictures are drawn on *picture sheets*, which have been adapted from XGAP graphic sheets, where $(0, 0)$ is the coordinate value of the top left-hand corner of the sheet. Pictures are drawn on individual picture sheets. Each picture is drawn on an individual picture sheet that has a unique identifier. On screen this is the name the user assigns to it, internally this is a number that XGAP assigns to the sheet. These picture sheets are produced with the command `PictureEditor` which requires a name for the sheet, it's width and the height. An optional entry to specify what presentation the picture on the sheet will represent can also be given. There are three drop down menus on SPICE sheets: the usual XGAP menu *"Sheet"*, the *"Picture Menu"* and the *"Shortcut Keys"* menu. The functions relating to the options of *Picture Menu* are discussed in Section 5.3. Figure 5.1 shows a screen shot of the options in *Picture Menu*. The *Shortcut Keys* menu provides a

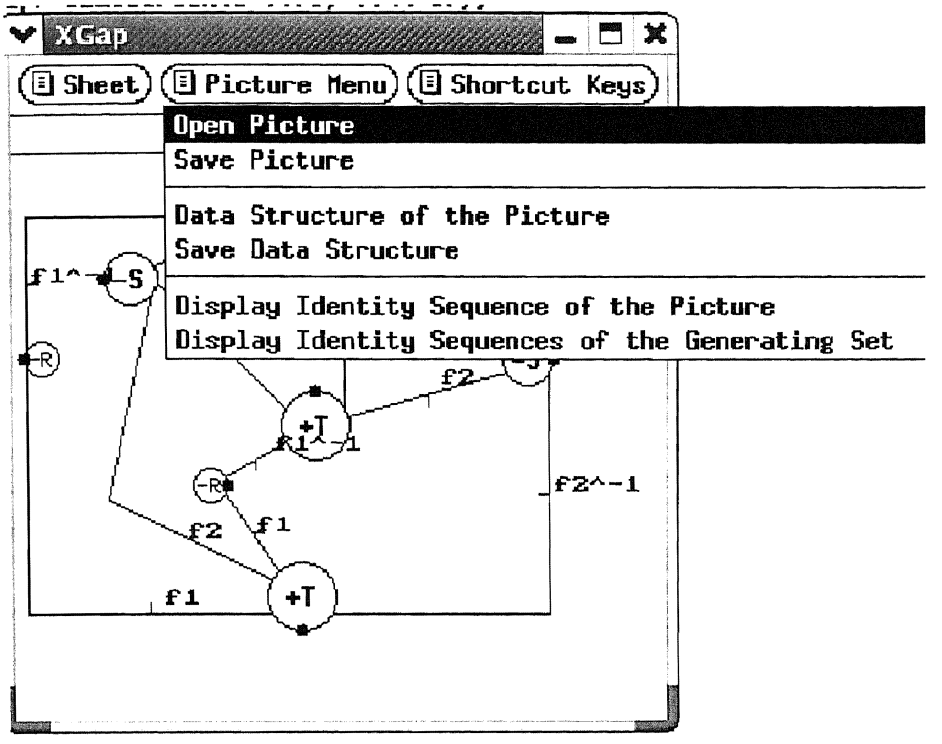


Figure 5.1: The *Picture Menu*

reference for shortcut keys and mouse clicks that manipulate picture objects (see Page 86 for more information).

The properties of a particular picture sheet are stored in a record, where the name of the record is the name of the sheet. This record has the following properties:

- `pdiscs` which is a list that stores the picture discs that have been drawn on the sheet.
- `arcs` which is a list that stores the arcs that have been drawn on the sheet.
- `RotationScheme` which stores the rotation scheme of the picture that is drawn on the picture sheet.
- `RelatorInformation` which stores the relator information of the picture that is be drawn on the sheet.
- `FpGroup` which gives the presentation that the picture is over.
- `GeneratingSet` if a picture is belongs to a generating set of pictures over `FpGroup`, this component gives the number of the picture in that set; otherwise it is set equal to false.

- `ExternalRegion` this stores the connection points of the external region of the picture.

The default attribute value for these components is set to false. Throughout this chapter is it assumed that all objects are created on a picture sheet with name *sheet.* The user can access a list of all discs and arcs on *sheet* with the functions `DisplayDiscs` and `DisplayArcs`. These functions produce the same output as calling `name!.pdiscs` and `name!.arcs`, but in keeping with GAP standard, record components are only used at a programming level. Similarly, functions have been implemented that display the presentation the picture is over, the data structure and the external region of a picture.

5.1 Creating Picture Objects

There are three types of picture objects: picture discs, basepoints and arcs. Unless otherwise stated, the term 'disc' will be used to refer to these picture discs and not the usual XGAP disc object. Picture objects can be created on screen in two ways: the relevant command can be typed into the XGAP window or, using certain keyboard and mouse shortcuts, the object can be produced directly on screen. These shortcuts (also referred to as 'callbacks' in [9]) consist of pressing either the shift or control key together with either the left or right mouse button; a 'shift left-click' therefore occurs when the left mouse button is pressed simultaneously to the shift key. The advantage of this latter option is that the object can be placed, or moved, exactly where the user desires without having to trifle with coordinate values.

The *Shortcut Keys* menu provides the user with a quick reminder of the shortcut key combinations. The user selects the option on the menu and the key combination appears in the XGAP window. Figure 5.2 shows the result of selecting the "Move Arc" option.

Creating basepoints and arcs is a more difficult task due to the fact that they need to lie exactly on the circumference of the disc. Arcs and basepoints can not be constructed until the disc that are incident on has been drawn. Assume a disc had coordinates (x_c, y_c) at it's centre and a radius r, then obviously finding a point, (x_1, y_1), on the boundary of this disc

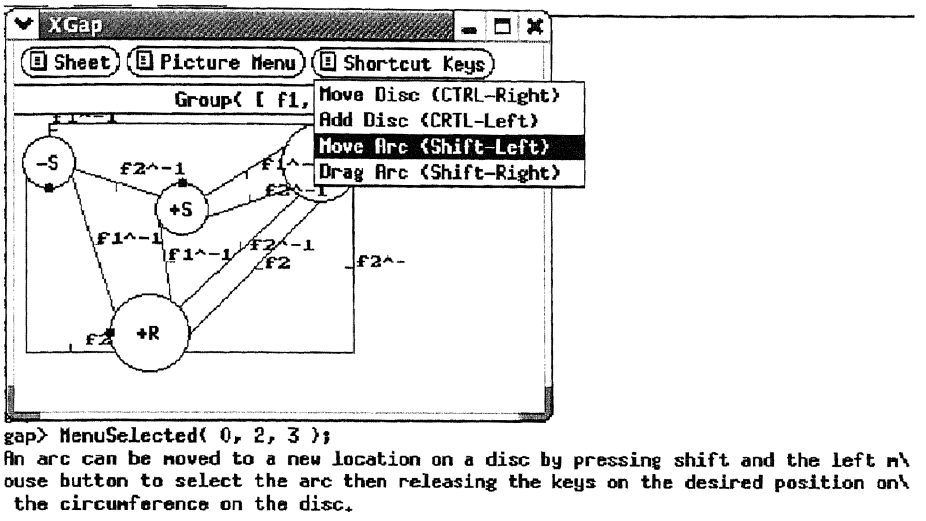


Figure 5.2: A screen shot of the *Shortcut Keys* menu

involves manipulating the equation of a circle, $(y_c - y_1)^2 + (x_c - x_1)^2 = r^2$. The problem is that both GAP and XGAP use atomic irrationalities to express square roots of rational numbers, which are of no use in coordinate geometry. Furthermore as XGAP sheets only entertain integer coordinates, a GAP function, `IntegerOfSquareRoot`, was written that took either an integer or a real number and called out to a C program, `SquareRoot`. `SquareRoot` computes the square root of the rational, rounds the answer to the nearest integer and writes this value to a file. `IntegerOfSquareRoot` then retrieves this number from the file and returns it as output. This is a relatively expensive, but essential, procedure and as such the number of times it is used to create and manipulate picture objects is kept to a minimum.

The properties of picture objects, as with XGAP graphical objects, are stored in records. Table 5.1 shows the components of the records for each picture object. Note that in keeping with GAP convention the American spelling for colour is used for the corresponding component names. By default, the boundary of picture discs and their labels are black and the lines of the arcs are blue with red arrows and red labels. The colour of these objects can be changed by the user by calling the XGAP `Recolor` function.

The XGAP functions `Circle` and `Text` are used to construct picture discs, where `Text` places the label of the disc at it's centre. The radius of a disc, Δ, is given by the value of $4 \times deg(\Delta)$. It was decided to keep the size of the disc in proportion to its degree, due to the

Object	Record Component Name	Stores
Picture Disc	`x`	the x-coordinate of the centre of the disc
	`y`	the y-coordinate of the centre of the disc
	`r`	the radius of the disc
	`label`	the label of the disc
	`basepoint`	the properties of the basepoint of the disc
Basepoint	`x`	the x-coordinate of the centre of the basepoint
	`y`	the y-coordinate of the centre of the basepoint
	`FirstArc`	the number of the first arc traversed after the basepoint, according to the orientation of the disc
Arc	`discs`	the coordinates of the centre of each disc
	`label`	the label of the arc
	`arc`	the arc numbers of the endpoints of each arc
	`lines`	the lines used to construct the arc
	`direction`	the direction of each connection point of the arc (1 denotes positive orientation, -1 otherwise)
	`arrow`	the properties of the arrow of the arc
	`color`	the colour of the arc

Table 5.1: A table to show the features of graphical picture objects

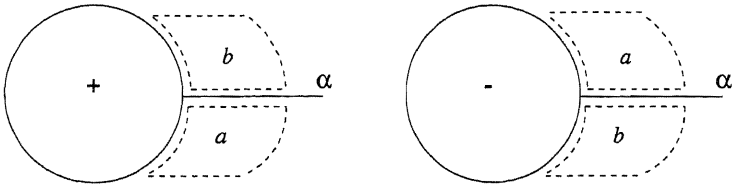


Figure 5.3: The areas a and b around a connection point

fact that a disc can have numerous arcs incident upon it. After several empirical, somewhat arbitrary, tests it was decided that this factor was the optimal choice for a disc's radii, as it produced the most visually appealing picture discs. The XGAP `Disc` function is used to create basepoints.

Arcs and arrows are constructed using the `Line` function of XGAP. Three of the components of the records for arcs: `discs`, `arc` and `direction` are lists of length two. Given an arc from Δ_1 to Δ_2, the first element of each list gives the relevant properties for Δ_1 whilst those for Δ_2 are given by the second entry of each list. Arrows are constructed concurrently with arcs. The arrow of an arc is given by a straight line transverse to the midpoint of that arc. It was deemed to be too computationally expensive to draw arrow heads, as this would involve calling the `IntegerOfSquareRoot` function three times. The following convention is therefore employed which utilises the `IntegerOfSquareRoot` function at most once.

Assume that there is a connection point of an arc α that is incident on disc Δ. Let b and a denote small areas before and after the connection point is encountered when traversing the circumference of Δ according to it's orientation, as shown in Figure 5.3. If the arrow is encountered in area b, (i.e. before the arc) when travelling around a disc this implies that that connection point has negative orientation, whilst encountering the arrow in area a (i.e. after the arc) means that the arc endpoint has a positive rotation.

A positive orientation means that the element which the connection point represents is given by the label of that arc. Similarly, a negative orientation indicates that the connection point represents the inverse of the label of the arc.

Example 5.1

Let x be the label of an arc incident on $\Delta_{i,j}$, shown in Figure 5.4. When traversing $\delta\Delta_i$, the arrow is met before the arc. The connection point of Δ_i therefore represents the element x^{-1}, whilst x is the label of the connection point incident on Δ_j. The second diagram of Figure 5.4 shows that x is the label of both of the connection points on Δ_i and Δ_j.

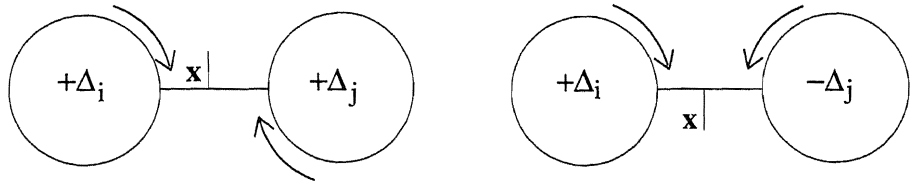


Figure 5.4: The technique used to depict arrows

A simple way to remember this convention for arrows is to think that encountering the arc before the arrow is much that same as travelling with the orientation of the arc, thus meeting the arrow first is 'going against the flow' i.e. a negative orientation.

5.1.1 Picture Discs

Picture discs are created using the function `PictureDisc`. This function takes as input the name of the sheet, *sheet* say, that this disc is to be drawn on; the coordinates of the centre of the disc; the length of it's radius and label. It then produces the required disc. The properties of the disc are then stored in the record for that disc.

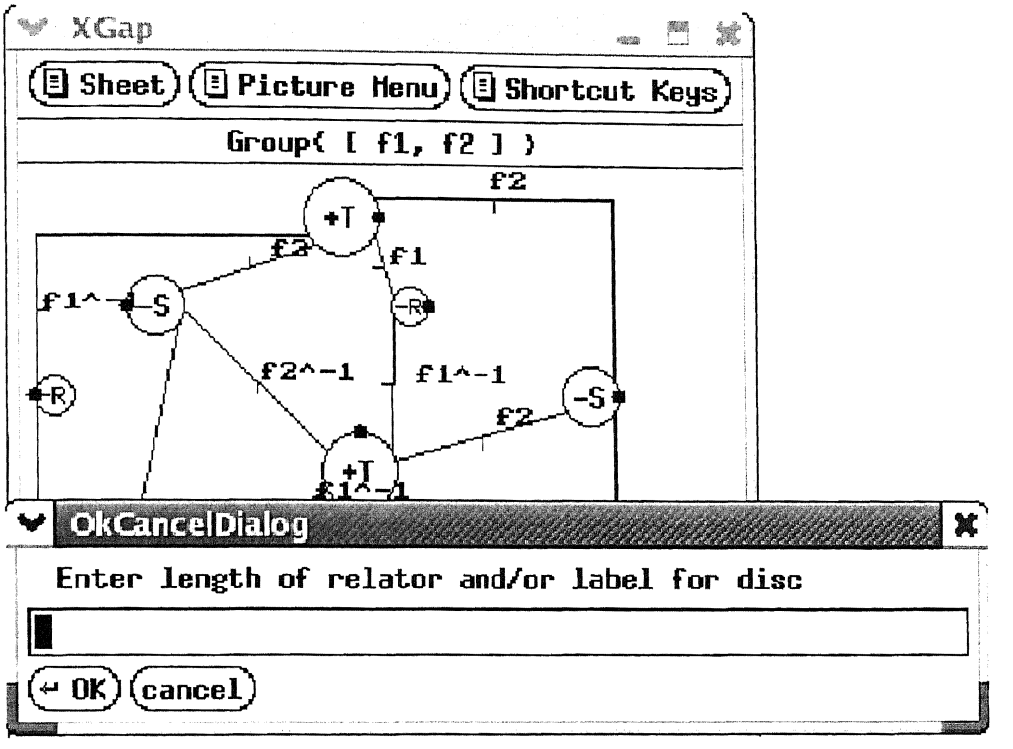


Figure 5.5: Adding a picture disc using keyboard shortcuts

Alternatively, the user can employ a shortcut technique by pressing the control key and the left mouse button at the point on the sheet where the centre of the disc is required.

This shortcut calls the function `AddPictureDisc` which produces an XGAP dialogue box, as shown in Figure 5.5. This dialogue box asks for the length of the relator and the label of the disc. These two pieces of information are entered into the dialogue box, separated by a comma. If the relator information for *sheet* has been previously declared the user only needs to enter the disc's label. In this instance, the records in the `RelatorInformation` component can be searched until the record containing that label is found. The relator corresponding to that label can then be obtained. In either case, the radius of the disc can be calculated by multiplying the length of the relator by four. `AddPictureDisc` then calls `PictureDisc` and the disc is drawn at the desired location of *sheet*.

5.1.2 Arcs

Arcs consist of a number of straight line segments; the general method to construct arcs involves determining the endpoints of, at most, 3 such lines. It is required that there is a

minimum of four vertical coordinate points between arc endpoints incident upon the same disc. This value of four was chosen because the radius of each disc is four times the degree of the disc. All arcs can therefore emanate from one quarter of the circumference of the disc, if necessary, whilst maintaining readability of arc labels and arrows. This readability factor was verified by performing several empirical tests on arcs incident upon discs with a variety of radii.

Let α be an arc from connection point n_1 of Δ_1 to connection point n_2 of Δ_2. The label, $W(\sigma_{\Delta_1,n_1})$ say, of the arc is stored as the element that a positive traversal of σ_{Δ_1,n_1} represents. Given that Δ_1 and Δ_2 have previously been drawn on a picture sheet, the concept adopted to construct α can be summarised as follows:

- Estimate suitable initial coordinate points, (x_k, y_k), for the arc endpoint on the circumference of Δ_k (for $k = \{1, 2\}$).
- If arcs have been constructed on Δ_k, ensure that placing the endpoints of α at the coordinate (x_k, y_k) adheres to the existing rotation scheme of Δ_k. If this is not the case, change the coordinate values of (x_k, y_k) accordingly.
- Ensure that drawing α with coordinate points $\{(x_1, y_1), (x_2, y_2)\}$ does not cause the arc to intersect Δ_1 and/or Δ_2 twice. If this is the case, a horizontal or vertical line, l_k say, from (x_k, y_k) is needed to avoid this. The endpoints of l_k are stored. Note that l_k will become a line segment of α.
- Draw α using the calculated values for the endpoints of the lines of α. The label of α is drawn at this stage.
- Find the direction of the connection points of α.
- Construct the arrow of α.

Determining Initial Coordinate Points for α

Given that $k = \{1, 2\}$, the idea is to estimate the values for (x_k, y_k) and then determine if these values contravene any existing rotation scheme of Δ_k. Clearly, the most suitable choice

of coordinates for the endpoints of α are the closest coordinates between the circumferences of Δ_1 and Δ_2. It was decided to select eight possible locations for these initial endpoints, as is illustrated in Figure 5.6.

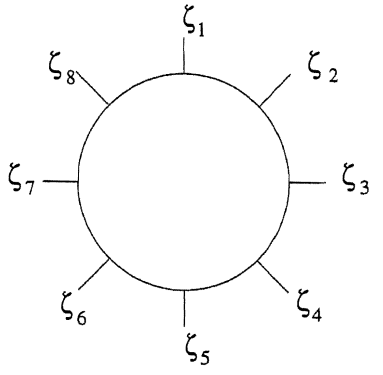


Figure 5.6: The possible initial locations for the endpoint of α

The criteria for finding the most suitable coordinates for these endpoints of α is therefore based upon the coordinates of the centre of Δ_k (x_{c_k}, y_{c_k}).

Example 5.2

If $x_{c_1} = x_{c_2}$ then the most appropriate method to construct α is between ζ_1 and ζ_5 of Δ_1 and Δ_2 (where ζ_1 and ζ_5 are positions on the circumferences of a disc, as shown in Figure 5.6). It obviously depends upon the values of y_{c_1} and y_{c_2} whether α is drawn between ζ_5 of Δ_1 and ζ_1 of Δ_2, as shown in Figure 5.7a, or between ζ_1 of Δ_1 and ζ_5 of Δ_2, as shown in Figure 5.7b.

Figure 5.7: a) Initial Endpoints when $y_{c_1} < y_{c_2}$ b) when $y_{c_1} > y_{c_2}$

If r_k depicts the radius of Δ_k then the coordinates of ζ_j, for $j = \{1, 3, 5, 7\}$ are not difficult to compute as they are given by a combination of the values of the centre of Δ_k and r_k. The y-coordinates of ζ_j, for $j = \{2, 4, 6, 8\}$, are found from the formula: $y_{c_k} \pm (r_k/2)$. For these latter points, the corresponding x-coordinate can be computed from the equation of a circle. Table 5.2 shows the criteria employed to select the appropriate location for the endpoints of

α.

Criteria	Point of $\delta\Delta_1$	Point of $\delta\Delta_2$
$x_{c_1} < x_{c_2}$ and $y_{c_1} < y_{c_2}$	ζ_4	ζ_8
$x_{c_1} < x_{c_2}$ and $y_{c_1} > y_{c_2}$	ζ_2	ζ_6
$x_{c_1} > x_{c_2}$ and $y_{c_1} < y_{c_2}$	ζ_6	ζ_2
$x_{c_1} > x_{c_2}$ and $y_{c_1} > y_{c_2}$	ζ_8	ζ_4
$x_{c_1} < x_{c_2}$ and $y_{c_1} = y_{c_2}$	ζ_3	ζ_7
$x_{c_1} > x_{c_2}$ and $y_{c_1} = y_{c_2}$	ζ_7	ζ_3
$x_{c_1} = x_{c_2}$ and $y_{c_1} < y_{c_2}$	ζ_5	ζ_1
$x_{c_1} = x_{c_2}$ and $y_{c_1} > y_{c_2}$	ζ_1	ζ_5

Table 5.2: The initial possible coordinates for endpoints of α

Ensuring that α Adheres to the Rotation Scheme

The second stage in the construction of α is to verify that constructing α with coordinates (x_1, y_1) and (x_2, y_2) on the boundary of Δ_1 and Δ_2 does not contravene the existing rotation scheme for each disc. Using Δ_1 as an example, where α has initial endpoint (x_1, y_1), the following possibilities need to be examined:

1. The arc α is the first arc to be drawn on Δ_1.
2. No arc has been drawn at the point (x_1, y_1) and
   (a) the arc α is the second arc to be drawn on Δ_1;
   (b) at least two arcs have been constructed upon Δ_1 before α has been drawn.
3. An arc has been constructed previously at the point (x_1, y_1).

Clearly, if α is the first arc to be constructed on $\Delta_{1,2}$ this step is not required and the appropriate coordinates for the endpoints of α are given by one of the initial values, $\zeta_j($ for $j = 1..8)$.

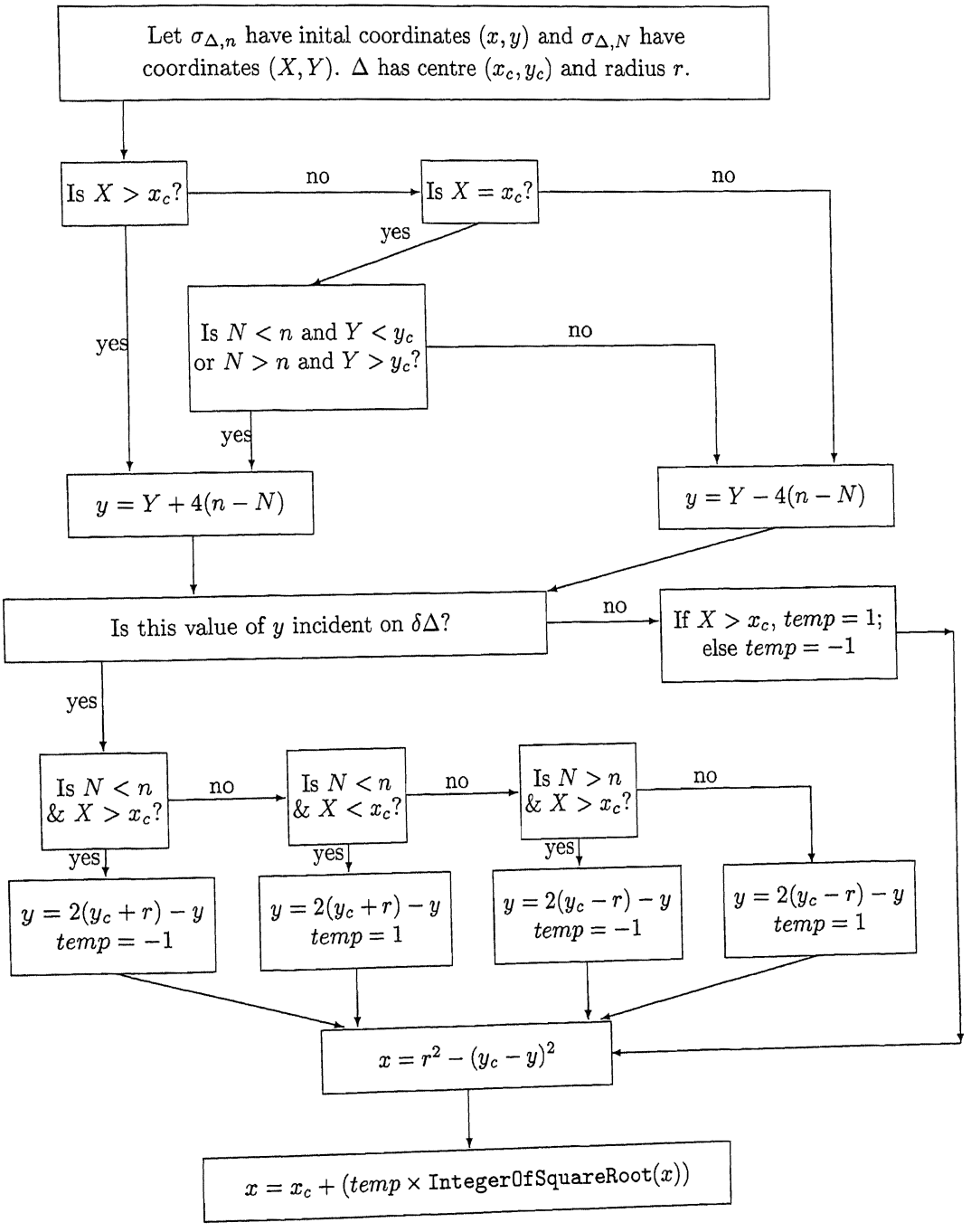
Assume that a possibility other than 1 arises. It needs to be ensured that the connection points with arc numbers less (respectively more) than n_1 occur before (resp. after) (x_1, y_1) around the circumference of Δ_1. This involves examining the arcs that have been constructed on the sheet that are incident on Δ_1. A list can be produced from this information, where the ith component of the list stores the coordinates of the ith numbered arc incident on

the boundary of Δ_1. An empty component indicates that the arc with that number has yet to be constructed. Performing this step also provides a way by which to check that the arc number n_1, of α, has not been previously constructed. If this is the case, an error is signalled. By examining this list, it can be determined whether an arc has previously been constructed at the point (x_1, y_1).

Given that the situation described in case 2(a) has occurred, let α_1 be the arc that has been constructed upon $\delta\Delta_1$ with arc number N. Now α_1 must be located at one of the positions of ϑ_j (for $j = \{1..8\}$), say at position (X, Y) on Δ_1, since one of the values given in Table 5.2 is always used for the first arc to be drawn upon a disc. All that remains in this situation is to ensure that there is sufficient (coordinate) difference between Y and y_1 to construct all the arcs of Δ_1 that occur from N to n_1. In other words, there must be $|4(n_1 - N)|$ points between y_1 and Y. If this is not the case then there cannot be the required minimum of 4 vertical units between each arc and hence the value for y_1 must be changed.

The method to change the value of y_1 is depicted in Figure 5.8. This diagram gives the procedure to recompute the initial coordinate point of an arc endpoint, for the general case. In other words, using the general notation that is used in Figure 5.8, let one connection point of the arc α be the nth arc incident on Δ. The initial coordinate values of this point have been calculated as (x, y). An arc has been drawn at the connection point $\sigma_{\Delta,N}$, at the point (X, Y). Given that $X = x$ and $Y = y$ or $|y - Y| < 4$ then the value of y is recalculated using this procedure.

Figure 5.8: The procedure to alter arc endpoints such that they adhere to the local rotation scheme of the disc



To explain the technique given in Figure 5.8 in more detail, the following example is given. Note that, for consistency, the notation used in Figure 5.8 is adopted in Example 5.3.

Example 5.3

It is known that either $X = x$ and $Y = y$ or $|y - Y| < 4$. Let $X < x_c$ and $N > n$. The new coordinate value for y is given by $Y - 4(n - N)$. This idea is illustrated in Figure 5.9.

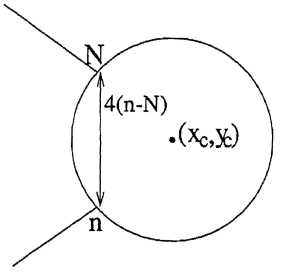


Figure 5.9: Determining the value of y

The possibility arises, however, that changing the original value of y by $4(n - N)$ points, gives a vertical coordinate point that does not lie on $\delta\Delta$. Hence, there is not sufficient vertical distance for both X and x be less than, or both greater than, x_c. Consequently if $X < x_c$ then x must be greater than x_c and vice versa.

Example 5.3 (cont.)

Assume that $y > y_c + r$, as is shown in the first diagram of Figure 5.10. In this example, $X < x_c$. Consequently, as the second diagram in Figure 5.10 depicts, $x > x_c$.

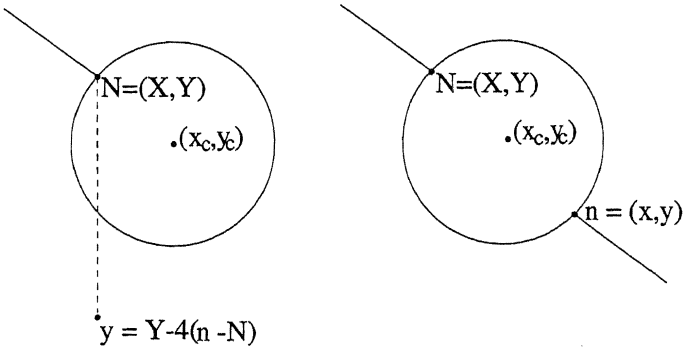


Figure 5.10: Determining the value of y from the position of arc N

Once the correct value for y-coordinate as been determined, the equation of a circle can now be used to compute the corresponding x value.

Returning back to the notation that has been used throughout this chapter, as detailed on Page 91, assume that more than one arc has been drawn upon Δ_1. The idea is to ensure that α is incident on $\delta\Delta_1$ between two relevant arc endpoints. The two arc's, α_1 and α_2 say, of Δ_1 whose arc numbers are closest to n are identified. For example, if the first, third and fifth arcs of Δ_1 have been drawn and $n_1 = 4$, the algorithm sets the third arc to be α_1 and the fifth to be α_2.

A possible drawing of α, α_1 and α_2 is depicted in Figure 5.11. The coordinates of α_k, for $k = \{1, 2\}$, are given by $(x_{\alpha_k}, y_{\alpha_k})$.

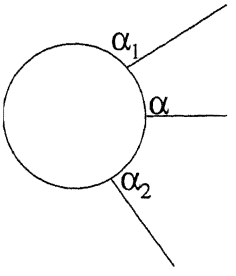


Figure 5.11: A possible drawing of α_1, α and α_2

To ensure that α does lie in-between α_1 and α_2 on $\delta\Delta_1$, the position of $(x_{\alpha_1}, y_{\alpha_1})$ and $(x_{\alpha_2}, y_{\alpha_2})$ relative to (x_{c_1}, y_{c_1}) could be considered. This gives a total of 15 possible combinations that would need to be examined.

Now, in the clockwise direction, α could either lie in-between α_1 and α_2, or in-between α_2 and α_1, depending on the arc numbers of α, α_1 and α_2. For each case there is therefore a further 2 possibilities to be examined. This gives a total of 30 cases to consider to determine if α does lie in between x_{α_1} and x_{α_2}. It would increase computation time to search through the 30 cases. An alternative approach was therefore adopted which renders a more efficient algorithm. This involves theoretically cutting $\delta\Delta_1$ at the point $(x_{c_1}, y_{c_1} - r_1)$ and transforming the circumference of Δ_1 into a horizontal line, l say, with an endpoint at $(x_{c_1}, y_{c_1} - r_1)$, as shown in Figure 5.12.

Each point on the circumference of Δ_1 can now be assigned a unique x-coordinate. Let `HLine` be a function such that `HLine`(x_1) gives the corresponding value of x_1 on l. The values of this step function, `HLine`(x_1), are depicted in Table 5.3.

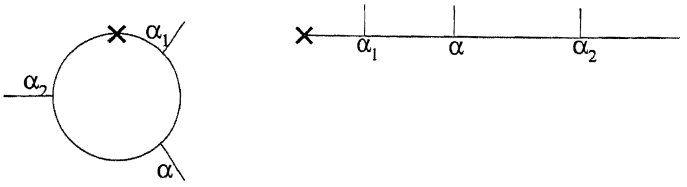


Figure 5.12: Illustrating the HLine function

Criteria	HLine(x_1)
$x_1 \geq x_{c_1}, y_1 < y_{c_1}$	x_1
$y_1 \leq y_{c_1}$	$2(x_{c_1} + r_1) - x_1$
$x_1 < x_{c_1}, y_1 < y_{c_1}$	$4r_1 + x_1$

Table 5.3: A table to show how the corresponding x-coordinates are calculated

Once the HLine values of x_1, x_{α_1} and x_{α_2} have been computed, the number of possibilities that need to be examined is reduced to 3:

1. $\texttt{HLine}(x_{\alpha_1}) < \texttt{HLine}(x_1) < \texttt{HLine}(x_{\alpha_2})$

2. $\texttt{HLine}(x_1) < \texttt{HLine}(x_{\alpha_2}) < \texttt{HLine}(x_{\alpha_1})$

3. $\texttt{HLine}(x_{\alpha_2}) < \texttt{HLine}(x_{\alpha_1}) < \texttt{HLine}(x_1)$

If one of the above equations are satisfied, then it is obvious that constructing α at position (x_1, y_1) does not contravene the local rotation of Δ_1. Note that, due to the way arcs are constructed, there must be sufficient vertical distance to draw α in-between α_1 and α_2 on $\delta\Delta_1$. The algorithm does check, however, if there is the required distance between α_1 and α and α and α_2; it may be the case that the value of y_1 needs to be changed by several points. If this occurs, the value of x_1 is recalculated accordingly.

If the corresponding value for x_1 does not lie in one of these three cases, α cannot be bounded by α_1 and α_2. The values for the endpoints of α on $\delta\Delta_1$ are then temporarily set to those of $(x_{\alpha_1}, y_{\alpha_1})$. The method given below is then utilised to compute the appropriate coordinate values for the connection points of α.

The final case to consider is that given by case 3. If an arc has been constructed previously at the position of (x_1, y_1) then the initial values for the endpoint of α on $\delta\Delta_1$ need to be altered. The idea is to use the arc that has been drawn at (x_1, y_1). The method depicted in Figure 5.8 is then employed to compute the new coordinates for the endpoint of α incident

on $\delta\Delta_1$.

The above process for determining the position of the endpoint of an arc from an initial estimation is repeated for the arcs on Δ_2. The coordinate values for the endpoints of α have now been determined such that they adhere to the rotation scheme for Δ_1 and Δ_2, and that they are the required distance apart on the disc's circumference.

Determining the lines of α

Constructing α with endpoints $\{(x_1, y_1), (x_2, y_2)\}$ could result in α intersecting Δ_1 and/or Δ_2 twice. This is not a useful situation as it reduces the readability of the overall picture; it not only looks confusing, as is shown in Figure 5.13a, but it is difficult for a user to interpret the direction of the arc's arrow. (In Figure 5.13a, all the arrows are intended to have a positive orientation.)

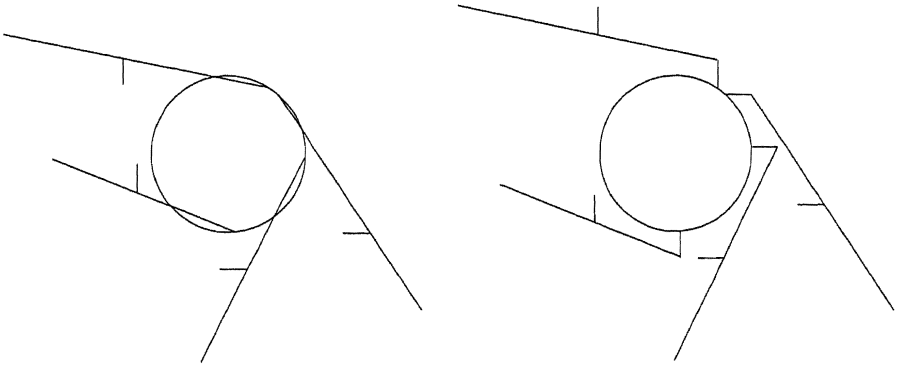


Figure 5.13: a) Intersecting Δ_1 twice b)The effect of constructing l_1

Assume that α does intersect Δ_1 twice. This problem is alleviated by constructing either a vertical or horizontal line, l_1, of length four units from the point (x_1, y_1). If $x_1 < x_{c_1}$ or $x_1 > x_{c_1}$ the endpoints for l_1 are given by $\{(x_1, y_1), (x_1 - 4, y_1)\}$ or $\{(x_1, y_1), (x_1 + 4, y_1)\}$ respectively. If $x_1 = x_{c_1}$ then the second endpoint of l_1 is given either by $(x_1, y_1 + 4)$ or $(x_1, y_1 - 4)$ depending upon whether y_1 is greater or less than y_{c_1}. Figure 5.13b shows the effect of adding these horizontal and vertical lines to the arcs in Figure 5.13a. The value of 4 was chosen as it is large enough to be seen but small enough to reduce the possibility that l_1 crosses the boundary of a disc adjacent to Δ_1.

Clearly the line from l_1 to the neighbourhood of Δ_2 may still intersect Δ_1 twice, the only way

this can be avoided is to increase the length of l_1. This could then cause the aforementioned problem: crossing another disc close to Δ_1. Nevertheless, even if the arc continues to intersect Δ_1 twice, constructing l does improve the readability of the arc. This is because the point on the circumference of the disc that α emanates from is clearer. Obviously, the same procedure can be applied to situations when α intersects Δ_2 twice.

The number of times a line intersects with an arc is found by determining the number of roots of the equation:

$$(m^2+1)x_{c_1}^2 + 2(m(c-y_1)-x_1)x_{c_1} + x_1^2 + (c-y_1)^2 - r_1^2 = 0$$

This is the solution of the equation of the line, $y_{c_1} = mx_{c_1} + c$, with the equation of the circle $(y_{c_1} - y_1)^2 + (x_{c_1} - x_1)^2 = r_1^2$. If α does intersect Δ_k (again, $k = \{1, 2\}$) twice, the endpoints for l_k are calculated. The coordinate points for the endpoints of each line of α are now known. Accordingly, each line of α can be constructed using the `Line` command of XGAP.

Constructing the label and arrow of α

Recall that an arc can represent both a generator of a presentation and the generator's inverse. It is impractical to store both of these elements, and indeed only one element can be written on-screen as a label of an arc. The label of α is therefore stored as the generator (or it's inverse) of $\mathcal{P}$ that is obtained from a clockwise traversal of σ_{Δ_1}.

Visually, the element that a connection point represents is determined by the label of the arc, the direction of the arrow and the orientation of the disc associated with the connection point. The XGAP `Line` function attaches a label around the neighbourhood of the line's midpoint (due to the implementation of this XGAP function, the exact location for the label depends upon the values of the endpoints of the line). The label of α is drawn close to the midpoint of the line of α that begins in the neighbourhood of Δ_1 and terminates in the neighbourhood of Δ_2. For instance, if both l_1 and l_2 exist, $W(\sigma_{\Delta_1,n_1})$ is placed in the neighbourhood of the midpoint of the line that connects l_1 and l_2.

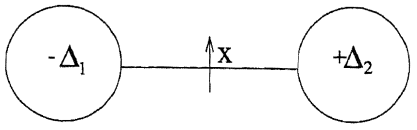


Figure 5.14: A possible drawing of α

The generator that each connection point depicts also needs to be obtained efficiently at a programming level. The difficulty that arises here is that if Δ_1 has an anti-clockwise orientation the label of α that is stored, $W(\sigma_{\Delta_1,n_1})$, is not the generator that σ_{Δ_1,n_1} depicts. Similarly, although $W(\sigma_{\Delta_1,n_1})^{-1}$ gives the generator that a clockwise traversal of σ_{Δ_2,n_2} represents, this may not be the required generator for that connection point.

Example 5.4

In Figure 5.14 the label of the connection point incident on Δ_1 is x. The label that is stored at a programming level, however, is x^{-1}.

This problem is overcome by storing the direction of the two connection points of each arc in a list of length two. This list then provides the direction of the arc. The first (respectively second) entry of the list, α`!.direction`, gives the direction of the arc with respect to a clockwise transversal of Δ_1 (resp. Δ_2). A positive orientation of the arc is denoted by 1, a negative orientation by -1. The label of the kth connection point of α, where $k = \{1, 2\}$, can therefore be computed by $W(\sigma_{\Delta_1,n_1})^{\alpha\texttt{!.direction[}k\texttt{]}}$.

Example 5.4 (cont.)

In Figure 5.14, the direction of α is given by the list $[-1, 1]$. This indicates that the generator that σ_{Δ_1,n_1} depicts is given by $W(\sigma_{\Delta_1,n_1})^{-1}$. Similarly, $W(\sigma_{\Delta_1,n_1})$ represents the label of σ_{Δ_2,n_2}.

It is not difficult to construct the direction of α, all that is required is to examine the orientation of Δ_1 and Δ_2. Table 5.4 demonstrates the four possible options for the direction of α. Directed arcs have now been created at a programming level. This method could be adapted to produce directed graphs in XGAP; a feature that has been requested. (See [62] for more information.)

The last part of the arc to be constructed is the arrow. As explained previously, the arrow

Orientation of Δ_1	Orientation of Δ_2	Direction of Arc
+	+	[1,-1]
−	−	[-1,1]
+	−	[1,1]
−	+	[-1,-1]

Table 5.4: The four possibilities for the direction of α

is depicted by a short horizontal or vertical line. The arrow is constructed transverse to a line, l, of α and has length five units. Since that the label of the arc is always given with respect to a clockwise traversal of σ_{Δ_1,n_1}, the direction of the arrow is always constructed with a positive orientation to Δ_1.

If l is either a vertical or a horizontal line, computing the coordinates for the arrows endpoints is not a complex procedure. Let the coordinates for endpoints of l be given by $\{(x_{l_1}, y_{l_1}), (x_{l_2}, y_{l_2})\}$. If l is horizontal the arrow is vertical. Once the midpoint, (x_M, y_M), of l has been found, the second endpoint of the arrow is given by $(x_M, y_M \pm 5)$. The decision to add or subtract five coordinate points to y_M depends upon the values of x_{l_1} and x_{l_2}, as is shown in Figure 5.15. For example, if $x_{l_2} > x_{l_1}$ (respectively $x_{l_2} < x_{l_1}$) then five units are subtracted (resp. added) to y_M.

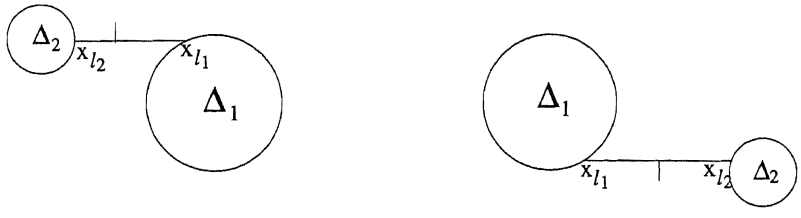


Figure 5.15: a) The arrow of α when $x_{l_1} < x_{l_2}$ b) The arrow of α when $x_{l_1} > x_{l_2}$

For vertical lines, the remaining endpoint of the arrow is determined by adding or subtracting five units from x_M. Again, in this situation only two possibilities need to be considered: if $y_{l_1} < y_{l_2}$ the remaining endpoint is $(x_M - 5, y_M)$, whereas if $y_{l_1} > y_{l_2}$ the coordinates for the arrow are $\{(x_M, y_M), (x_M + 5, y_M)\}$.

Figure 5.16: a) A drawing of α b) Constructing the arrow by comparing x_{c_1} and x_{l_1}

It may seem surprising that the position of (x_{l_1}, y_{l_1}) and (x_{l_2}, y_{l_2}) are considered when determining the coordinate point for the remaining endpoint of the arrow, rather than (x_{l_1}, y_{l_1}) and (x_{c_1}, y_{c_1}). If (x_{l_1}, y_{l_1}) and (x_{c_1}, y_{c_1}) are compared, consider the case when the arrow is vertical and $x_{l_1} < x_{c_1}$. The coordinates of the endpoints of the arrow would be given by $\{(x_M, y_M), (x_M, y_M - 5)\}$.

Consider the diagram in Figure 5.16a, where $x_{l_1} < x_{c_1}$. Constructing the arrow by comparing x_{l_1} and x_{c_1} gives the coordinates for the endpoints as $\{(x_M, y_M), (x_M, y_M - 5)\}$, as shown in Figure 5.16b. When traversing Δ_1 in Figure 5.16b in a clockwise direction, however, the arrow is encountered before the arc! This contradicts the convention that the arrow should have a positive orientation with respect to a clockwise traversal of Δ_1. Nevertheless, if the location of x_{l_2} is compared to x_{l_1} as opposed to x_{c_1}, the arrow is given the coordinate points $\{(x_M, y_M), (x_M, y_M + 5)\}$, as required. The same argument can be applied when constructing horizontal arrows.

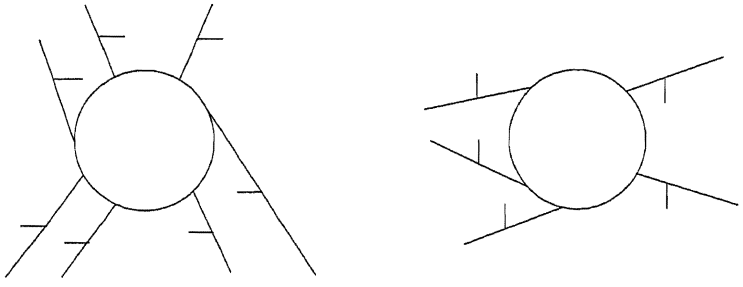


Figure 5.17: a) Drawing arrows when $m > |1|$ b) Drawing arrows when $m < |1|, m \neq 0$

Determining the endpoints of arrows for arcs that have gradients not equal to one or zero is more difficult. After several trials it was found that if the gradient, m, of l is less than $|1|$ (and not equal to zero) the most appropriate technique to visualise the arrow was by using a vertical line. In other words, the coordinates for the endpoints of the arrow would given by $\{(x_M, y_M), (x_M, y_M \pm 5)\}$. The decision to add or subtract five units from y_M again depends upon whether x_{l_1} is greater or less than x_{l_2}, as is shown in Figure 5.17a.

Similarly, for $m > |1|$ the coordinates for the endpoints of the arrow are $\{(x_M, y_M), (x_M \pm 5, y_M)\}$. If the value of y_{l_1} was greater (respectively less) than that of y_{l_2} then the coordinates for the endpoints of the arrow are given by $\{(x_M, y_M), (x_M + 5, y_M)\}$ (resp. $\{(x_M, y_M),$

$(x_M - 5, y_M)\}$). This logic is shown in Figure 5.17b.

Given that one of the coordinates for the arrow is the midpoint of α, Table 5.5 summarises the criteria employed to calculate the coordinates for the remaining endpoint.

Gradient of l	Criteria	Coordinates
0	$x_{l_1} < x_{l_2}$	$(x_M, y_M + 5)$
	$x_{l_1} > x_{l_2}$	$(x_M, y_M - 5)$
1	$y_{l_1} < y_{l_2}$	$(x_M + 5, y_M)$
	$y_{l_1} > y_{l_2}$	$(x_M - 5, y_M)$
$\|m\| < 1, m \neq 0$	$x_{l_1} > x_{l_2}$	$(x_M, y_M + 5)$
	$x_{l_1} < x_{l_2}$	$(x_M, y_M - 5)$
$\|m\| > 1$	$y_{l_1} > y_{l_2}$	$(x_M + 5, y_M)$
	$y_{l_1} < y_{l_2}$	$(x_M - 5, y_M)$

Table 5.5: The criteria to determine coordinates of arrows

This general procedure for constructing arrows is implemented in the function `Arc`. The input parameters for this function are as follows: the sheet, the picture discs, Δ_1 and Δ_2, a list for the arc numbers of the connection points, $[n_1, n_2]$, and the label of the connection point σ_{Δ_1, n_1} (again, with a respect to a clockwise traversal of Δ_1). Once the arc has been created, it's properties are stored in the record for that arc.

Arcs can also be drawn with the function `ArcWithCoords` and `ArcWithBends`. The first of these functions enables the user to specify the coordinate points of (x_1, y_1) and/or (x_2, y_2). Due to the fact that XGAP can only work with integer coordinates it can be cumbersome to find a coordinate point that lies exactly on the circumference of a disc, thus a small margin of error is allowed for the values of (x_j, y_j), for $j = \{1, 2\}$.

If (x_j, y_j) does not lie on a disc's boundary `ArcWithCoords` searches through a small neighbourhood of points around (x_j, y_j) to find a point on the circumference. If such a point cannot be found an error is signalled and the arc is not drawn. If a point can be found, however, then the function uses that value for the location of an endpoint of the arc. This function ensures that placing the arc endpoints at these coordinate values does not transgress the existing rotation scheme for Δ_1 and/or Δ_2. `ArcWithCoords` constructs arcs by employing the same technique as the general method described above.

`ArcWithBends` takes a collection of coordinate points (including (x_1, y_1) and (x_2, y_2)) which form the points where 'bends' occur in the arc; these bends are created by drawing separate

lines whose endpoints correspond to the given coordinate points. It is thought that this latter function will not be used directly by a user but is intended more for the automatic visualisation of pictures, implemented in the `DrawPicture` function described in Chapter 6. For this reason, this function does not check that the endpoints of the arc concur with the existing rotation scheme of the discs they are incident upon. This fact would be known by `DrawPicture`, and so to retest this would be unnecessary. Alternatively, if the user used `ArcWithBends` directly and the rotation scheme of a disc was contravened, the function `EnterPicture` (described in Section 5.3) would highlight the error.

5.1.3 Basepoints

Basepoints are XGAP discs of radius three and are drawn on the boundary of an existing disc with either the functions `Basepoint` or `PlaceBasepoint`. The first function automatically computes the location of the basepoint, whilst the latter enables the user to place the basepoint at a certain point on the boundary of the disc.

The two functions take as input the picture sheet, *sheet*, and disc, Δ say, that the basepoint is to be drawn on. `PlaceBasepoint` also takes a list of length two which gives the arc numbers that the basepoint must be placed between. With this latter function *sheet*`!.arcs` is examined to find the two arcs incident upon Δ. This gives two sets of coordinates, (x_1, y_1) and (x_2, y_2) say, for the endpoints of the relevant arcs incident upon Δ.

The procedure to find the midpoint between these points involves using the `HLine` function which theoretically cuts the boundary of Δ and transforms it into a horizontal line, as described in Section 5.1.2. The corresponding values of $x_{1,2}$ on this line are computed according to the method given in Table 5.3. The midpoint of the boundary between (x_1, y_1) and (x_2, y_2) is given by

$$\texttt{HLine}\left(\texttt{HLine}(x_1) + \frac{\texttt{HLine}(x_2) - \texttt{HLine}(x_1)}{2}\right)^{-1}.$$

The y-coordinate of this midpoint is calculated using the equation of the circle. The centre

of the basepoint is then placed at this position.

The `Basepoint` function can only be called when all arcs incident upon Δ have been constructed. The relator that Δ represents is determined either by referring to the relator information of that picture or, if this has not been defined, by calling the `DataStructure` function. `DataStructure` computes, and stores, the relator information and rotation scheme of a picture from it's drawing (see Section 5.3). Note that in the latter case, all arcs of the picture must be constructed to obtain the data structure of the picture.

This step was seen as a more efficient approach to adopt rather than requiring that the relator be given as an input parameter for `Basepoint`. Firstly, it ensures that the picture is a picture over a group presentation; if the data structure cannot be computed the drawing must not depict a picture over that presentation. Moreover if the data structure of the picture does not exist, the drawing on the sheet is nothing more than a collection of picture objects. At a programming level this drawing only becomes a picture when its data structure has been stored. It is therefore advantageous to calculate and store the data structure of a picture at the earliest moment.

Once the relator that Δ represents has been identified, `Basepoint` implements a procedure similar to `CheckDiscs` (as described in Chapter 2), to find the location of the basepoint. This method involves searching through *sheet*`!.arcs` for all, n say, arcs that are incident on the disc. The generator that the endpoint of these arcs depicts is given by taking the label of the arc and multiplying it by that connection point's direction. The disc is traversed according to its orientation. The labels of the connection points are stored as they are encountered and a word, $W(\sigma_{\Delta,1})W(\sigma_{\Delta,2})\cdots W(\sigma_{\Delta,n})$, is produced. This word is permuted until it equals the desired relator, say $W(\sigma_{\Delta,i})W(\sigma_{\Delta,i+1})\cdots W(\sigma_{\Delta,i-1})$. One possible location of the basepoint is then between the arcs numbered by $W(\sigma_{\Delta,i-1})$ and $W(\sigma_{\Delta,i})$. Again the procedure to find the midpoint between two coordinates on the circumference of a disc is employed and the basepoint is produced with its centre at this position.

5.2 Manipulating Picture Objects

This section describes how picture objects can be manipulated on-screen. Picture discs, basepoints, arcs and arrows can all be deleted with the `Delete` procedure. This is an adaption of the XGAP `Delete` function which destroys the object from the sheet. If a line of an arc is to be deleted and the arc's arrow is incident upon that line, the arrow and the label of the arc are also deleted.

The XGAP `Highlight` function has also been altered to apply to picture discs and arcs. This function changes the colour of the circumference of discs and arc lines to green and increases their width. As will be shown, this is a useful visual aid when an object is being manipulated.

5.2.1 Picture Discs

Picture discs can be moved to a different position on the sheet with the function `MoveDisc`. Given an existing disc, Δ, and a set of coordinates for its new centre, `MoveDisc` deletes Δ from it's position and redraws it at the new location. The basepoint and all the arcs that were incident on Δ are deleted from their original position and redrawn accordingly.

The arcs are redrawn by finding all arcs on the sheet that emanate from Δ. For each arc, the other disc that it is attached to, the label and the arc numbers of the endpoints are stored. The arc is then deleted and redrawn with the `ArcWithCoords` function. Basepoints are easily redrawn by saving the property `FirstArc`, deleting the original basepoint and constructing it at the new location using `PlaceBasepoint`.

Discs can also be moved to a new location on the picture sheet by pressing the control key and the right mouse button directly on the disc and dragging it to a position. When a control and right mouse click occurs, at position (X, Y) on the sheet say, this calls `PropertiesOfDisc`. This then finds the disc, Δ, whose circumference binds (X, Y). The boundary of Δ is highlighted. When control-right click is released at a new location on-screen, this point now becomes of the centre of Δ. `MoveDisc` is called and Δ, it's basepoint and the arcs that are

incident upon it, are redrawn accordingly.

5.2.2 Arcs

Arcs can be manipulated in three ways: "bends" in arcs can be created, the endpoint of an arc can be moved around a disc and the position of a line of an arc can be changed. The first two of these functions can be called by employing keyboard shortcuts, where a line of an arc is selected by placing the cursor over it and pressing certain keys and mouse buttons. It can be quite difficult to place the cursor exactly at a point on the line. If the selected point is not incident on a line, a small neighbourhood of coordinate points around the point is searched to locate the required line.

Let α be an arc with a line, l, which has endpoints at $\{(x_1, y_1), (x_2, y_2)\}$ with (x_b, y_b) a point on the sheet. A bend of α is produced at (x_b, y_b) by deleting l and creating two lines with endpoints $\{(x_1, y_1), (x_b, y_b)\}$ and $\{(x_b, y_b), (x_2, y_2)\}$. If the arrow of α was incident upon the original line l, this is also deleted with l and redrawn at the midpoint of the first of the two new lines. The procedure is called with the shortcut procedure 'shift right click'. The user selects a point of α with the shift key and right mouse button. This activates `UserDragArc` which searches through the lines of arcs on the sheet until the line that the user has selected is found; this line is then highlighted. The user drags the cursor to the desired point for (x_b, y_b) and releases 'shift right click'. The required bend of α is then constructed.

Arc endpoints can be moved around the circumference of a disc by using the shortcut 'shift left click'. The user selects an arc, α, by pressing the shift key and the right mouse button and drags a connection point of α to a new point on the boundary of the disc, Δ say, that it is incident upon. Once α has been selected the algorithm searches through all arcs on the sheet until α is found.

Again, α is highlighted. The point at where the user released 'shift right click', (X, Y) say, becomes a possible new endpoint for α. The picture discs that α is incident upon are found by the function `PropertiesOfDisc`. The label and arc numbers of α are stored and α is deleted from the sheet. `ArcWithCoords` is used to redraw α with an endpoint at (X, Y) on

Δ. If (X, Y) does not lie in a small neighbourhood of $\delta\Delta$, or constructing the endpoint at (X, Y) contravenes the existing rotation scheme for Δ, `ArcWithCoords` signals the error and does not draw the arc here. In this situation the arc is redrawn at the original position.

The position of the endpoints of a line, l, of an arc α can be changed using the function `MoveArcLine`. This requires α, the line l and the new coordinate values for the endpoints of l as input parameters. This procedure deletes l from the sheet and redraws it accordingly. Furthermore, if the arrow of α was incident upon l originally, then the arrow is also redrawn. Unfortunately this function could not have a shortcut procedure associated with it. This is due to the fact that XGAP only admits four callback functions. These callback procedures have been assigned to the functions that, it was felt, would be the most useful for the user.

5.2.3 Basepoints

The function `ChangeBasepoint` enables a user to modify the position of the existing basepoint on a disc, Δ. Let l be the first arc that occurs after the basepoint, according to the orientation of Δ. The same concept as that utilised by `Basepoint` is employed to change the basepoint. The word ω is obtained by reading the labels of the connection points of the arcs around Δ, according to its orientation and the direction of the line, beginning from arc l. This word is permuted until it is again identically equal to the relator of Δ. If this does not occur, a message declaring that there is only one possible location of the basepoint on Δ is produced. If Δ has more than one basic corner, however, the original basepoint is deleted and a new basepoint constructed at the next possible position from l. This procedure can be iteratively called so that the basepoint can be drawn at any basic corner of Δ.

5.3 Capabilities of the Editor

There are three main parts to the features of SPICE. The first two sections are related to the picture sheet menus: one enables a user to gain information about the picture, or set of pictures, being depicted, and another allows pictures to be saved, either as they appear on

screen or as a postscript file. The third part of SPICE contains functions that are utilised to provide a deeper internal structure to the picture that the sheet represents.

5.3.1 The Picture Menu

Let **P** be a picture drawn on a picture sheet, *sheet*. The three main pieces of information about **P** that can be obtained directly from *sheet* are the rotation scheme, the relator information and the identity sequence of **P**. If **P** is part of a generating set of pictures then the identity sequences for all pictures in the generating set can also be produced.

The *Picture Menu* contains the entries **Data Structure for Picture** and **Save Data Structure**. **Data Structure for Picture** displays the data structure of the picture in the XGAP window, whilst **Save Data Structure** asks for a filename where the data structure can be saved. If a picture does not have a rotation scheme or a list for the relator information associated with it, selecting either of these option calls the function `DataStructure`. This function computes the rotation scheme and relator information for the picture simultaneously, and is described below.

The rotation scheme for **P** from its drawing is computed using the following method. For each disc, Δ_i, the length of the disc's radius is divided by four to give the number of components in the list for the local rotation of Δ_i. Let α be an arc from Δ_i to Δ_j, with connection points $\{\sigma_{\Delta_i,n}, \sigma_{\Delta_j,m}\}$. The nth entry for local rotation of disc Δ_i contains the list: $[\Delta_j, m, \alpha!.\texttt{label}^{\alpha!.\texttt{direction}[1]}]$. Obviously, the connection point associated with $\sigma_{\Delta_j,m}$ can also be determined easily from this information. This process is repeated for all arcs incident upon Δ_i, and all discs until the rotation scheme, Π, for **P** is obtained.

For each relator of the presentation, a record can be constructed with the component `Relator` set equal to that relator. These records form the basis for `RelatorInformation`. From Π, the relator that a disc, Δ_i say, represents can be computed from the product of the generators in Π_{Δ_i}, using a similar procedure to `CheckDiscs`. The record for this relator is found by searching through `RelatorInformation`; if the label of the disc is positive (respectively negative) then i is appended to the `PositiveDiscs` (resp. `NegativeDiscs`). If the

`Label` component for that relator has not been stored previously then it is obtained from the label for disc i. Iterating this procedure for all discs in $\mathbf{P}$ gives the `RelatorInformation` structure, as required.

The identity sequence a picture represents can be obtained by selecting 'Identity Sequence of Picture' from the *Picture Menu*. This calls the function `VisualIdentitySequence`. This is a similar function as `IdentitySequenceFromPicture`, described in Chapter 3, however here the external region is the visualised external region and not the region with the largest number of connection points. The identity sequence that is produced therefore corresponds to the displayed picture. If no exterior region has been declared in the properties of *sheet*, the algorithm again chooses that region with the largest number of connection points.

The identity sequences for all the pictures in the generating set can be obtained by selecting **Display Identity Sequences of Generating Set**, which calls the `IdentitySequenceFromPicture` procedure.

Example 3.2 (cont.)

A presentation for the group Q_4 is given by $< a, b; a^{-2}b^4, a^{-1}bab >$. A picture over this presentation, together with output produced by calling the 'Identity Sequence of Picture' command is given in Figure 5.18, were $f1 = a$ and $f2 = b$.

5.3.2 Opening and Saving Pictures

Pictures can be saved in two ways: as objects on picture sheets or as a postscript file. The first of these option enables a user to reopen existing pictures on picture sheets. These pictures can then be manipulated or information about the presentation they represent can be obtained.

Saving pictures as postscript type enables the complete picture to be seen easily, as only sections of large picture sheets can be seen on-screen at once. Moreover, it counteracts the problem of using a drawing package to reproduce the picture for publications: as it can be time consuming to draw pictures by hand, it is an even bigger task to make use of various drawing packages to produce these pictures. Furthermore using SPICE to create pictures

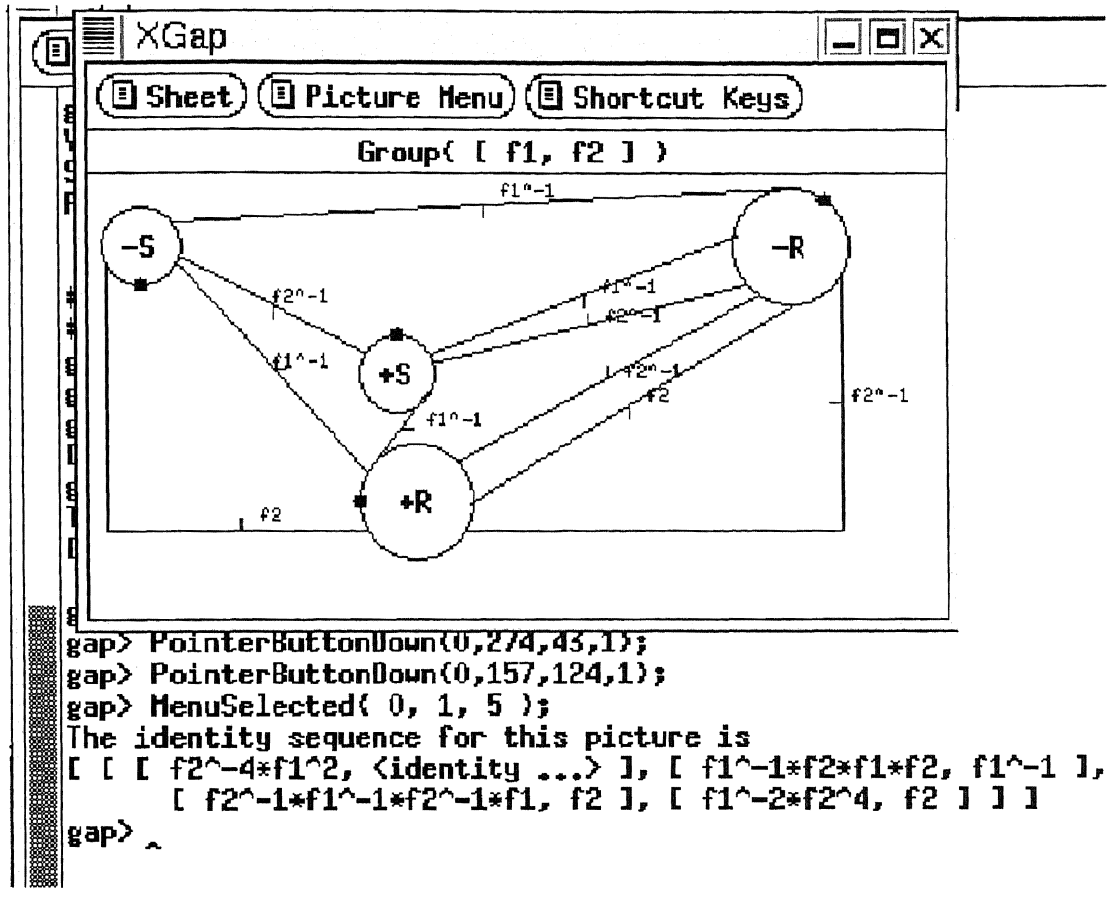


Figure 5.18: The output from selecting the 'Identity Sequence of Picture' option

ensures that there are no drawing errors in the picture.

The technique adopted to save picture objects on graphic sheets is to store the information used to create each object in a text file. For instance, discs are created with coordinates for the centre of the disc, a radius and a label. For each disc this information is stored in a list and appended to the list that contains the lists for all discs in that picture. There is also a list of lists for arcs and basepoints. This information can be acquired easily from the properties of these objects. The label of the arcs need to be transformed into a string, in order that the text file the information is saved in can be read by GAP. The external region is also transformed into string and saved into a list.

The remaining information needed to save a sheet is the name, height and width of the picture sheet itself, again this stored as a list. Generated sets of pictures can be saved into the one file by creating separate arrays of lists for each picture. This is useful when opening

saved pictures as SPICE inherently knows that the pictures are part of a generating set and the property `GeneratingSet` can be automatically computed for each picture sheet in the set.

Pictures saved in this fashion can be opened with the function `OpenPictureEditor`, which takes the name of the text file the pictures are saved in and the group presentation they represent. The text file is read as a function and for each array of lists a separate picture sheet is created. The functions `PictureDisc`, `ArcWithBends` and `Basepoint` are used to reconstruct the picture objects. The underlying properties of each sheet are stored: `FpGroup`, `GeneratingSet` and `ExternalRegion`. The function `DataStructure` computes the rotation scheme and relator information of that picture.

Saving pictures as postscript files involves creating postscript strings for picture discs, basepoints and arcs. This technique utilises the existing postscript strings for XGAP objects: the postscript string for picture discs is the XGAP string for circles, which in turn stores the label of the disc; basepoints are given by the postscript string for XGAP discs and arcs can be represented by postscript strings for lines (again this automatically stores the arc's label).

5.3.3 Storing the Properties of a Picture

Functions have been implemented that enable the user to store properties of a picture they have created. At a programming level, these functions give a structure and meaning to the picture. These functions are described as follows:

`FpGroupOfPicture` This enables the user to store the presentation that the picture represents.

`EnterGeneratingSet` This function stores a set of pictures as a generating set of a presentation. Each picture sheet is assigned a number, which corresponds to the number of the picture in the generating set. The rotation scheme and relator information of the pictures are also computed and stored using `DataStructure`.

`RelatorLabels` allows the user to assign labels directly to relators. This is useful when drawing discs on-screen using keyboard shortcuts, as now only the label itself needs to be entered.

`EnterPicture` If a collection of arcs and discs have been drawn directly on-screen this function investigates if they form a picture over a presentation by constructing basepoints on each disc. If the user has also drawn the basepoint on each disc manually (using `PlaceBasepoint`), the function checks that the word obtained by traversing the boundary of each disc, from the basepoint, gives a relator of the presentation (i.e. it calls the `CheckDiscs` function). In both cases, if this test is passed, then the data structure for that picture is constructed and saved together with the properties of the sheet.

`EnterPictureWithPlanarityTest` This function is the similar to `EnterPicture` however, after the data structure for the picture has been constructed, a planarity test is performed on the picture. The reason for this is that a planar picture may look non-planar due to the way in which arcs are drawn. The manipulation procedures described in Section 5.2 can be employed to rectify this problem, but a user may wish to know if a complex picture is planar before attempting to modify it.

5.4 Database of Pictures

The editor also contains a database of generating pictures for presentations of the 45 non-abelian groups of order less than 32. These pictures have been constructed using the `DrawPicture` program with the identity sequences provided by [24]. These 45 presentations are given in Appendix C. The pictures in the database have been modified, using the functions described above, for readability.

Each presentation has two files associated with it: one file gives the data structure for the presentation and the second stores the graphical information about each picture object. Both files are called by the group that the presentation represents or by Pn, if the group

defined by the nth presentation in Appendix C does not have an associated name.

The files that contain the data structure of the picture have the extension `'.pic'`, whilst the extension `'.sp'` denotes files that store the graphical information. The data structure files store the presentation, the identity sequences that generate π_2, the relator information and rotation scheme for each generating picture. These files were constructed using the `PictureFromIdentitySequences` program, with the `Label` component of each record in relator information inserted manually.

The database provides a library of generating sets which can be referred to when investigating properties of π_2 and be used to test new ideas and geometric algorithms. The idea is that if a user 'discovers' a new set of generating pictures for a presentation, this set can be submitted to an email forum and added to the SPICE database. In this way the knowledge of 3-presentations increases. The aim of the database, as with all SPICE software, is to provide tools which could assist in computing a deeper understanding of the second homotopy module.

5.5 Conclusions

The chapter has presented the features of SPICE and it's graphical picture objects. It is felt that SPICE is a useful picture drawing editor: it enables a user to draw, and manipulate pictures manually on-screen and it has standard drawing editor functionality such as opening and saving pictures. Furthermore, it has functions associated with it that enable a user to see properties of a picture that is displayed on a picture sheet.

Notwithstanding that SPICE is a worthwhile tool, there are some areas were it could be improved. The current method of drawing arcs could be modified such that arc crossings are completely avoided, for instance. As will be shown in the next chapter, however, this is an NP-hard task. Furthermore, it would be visually more appealing, and increase the readability of the picture, if an approximated spline was used to construct arcs, as opposed to a collection of straight line segments. Being able to draw arcs directly on-screen and

increasing the number of short-cut functions would also make this editor more user-friendly.

It is hoped that these modifications, along with new algorithms and generating pictures, will be added to SPICE in the future so that it becomes a useful and practical software tool for studying the second homotopy module.

Chapter 6

The Visualisation of Pictures

This chapter presents a graph drawing algorithm that visualises pictures from a given representation. SPICE picture sheets, described in Chapter 5, are used to display these pictures. Drawing planar graphs is a fundamental graph drawing problem (see [15] for example) with an NP-hard complexity [31]. Existing planar graph drawing algorithms are described in Section 2 of this chapter. The majority of these existing algorithms, however, are only valid for simple graphs. This algorithm uses the basic idea of these techniques and expands them to visualise pictures. It transpires that the method presented here does not always produce perfect picture drawings as arc crossings may still be present. Nevertheless, in the worse case scenario, it does provide an initial drawing of a picture. The techniques described in Chapter 5 can be employed such that the resultant picture can be easily manipulated to improve readability and look 'aesthetically pleasing'. A theoretical algorithm has been developed which would improve upon the picture drawing algorithm, but as is shown in the final section of this chapter, would be extremely difficult to implement.

6.1 Existing Graph Drawing Algorithms

A graph can have infinitely many drawings associated with it. The aim of graph drawing algorithms is to produce aesthetically pleasing, readable graphs. The *readability* of a graph

is the "capability of conveying the meaning of the diagram quickly and clearly" [15]. For instance, it is much easier to obtain information from a planar, straight line graph that one that contains many bends and edge crossings.

The problem of producing such useful graph drawings has been examined extensively; this is shown in a bibliography of graph drawing algorithms [15] which cites over 300 references. This diversity comes from the fact that there is a large number of applications where producing graph drawings automatically is a useful tool. In computer science such applications include artificial intelligence (knowledge-representation diagrams), databases (entity-relationship diagrams), decision support systems (PERT networks, activity trees), information systems (organisation charts), real-time systems (Petri nets, state-transition diagrams) and software engineering (data flow diagrams, subroutine-call graph, program nesting trees, object-orientated class hierarchies). Applications can also be found in other areas of science: evolutionary trees in biology, map schematics in cartography and molecular drawings in chemistry, to name but a few.

For each graph type there is a set of *graphic standards* which provide the criteria that the graphs in this set need to satisfy. These graphic standards include the *drawing convention* and *aesthetics* of the graph. The drawing convention gives the type of graph drawing required whether it be that each edge in the graph is drawn as:

- a straight line segment (this produces a straight-line graph drawing);
- a polygonal chain (this produces a polyline graph drawing);
- an alternating collection of horizontal and vertical line segments (this produces an orthogonal drawing). This graphical standard cannot be utilised to produce a picture drawing due to the fact that only graphs with vertices of degree at most four admit a planar orthogonal drawing ([16], Chapter 5).

The other main drawing conventions are grid drawings, where vertices, edge bends and edge crossing have integer coordinates, and planar drawings. (See [16], Chapter 2 for a detailed description of all of these conventions.)

The aesthetics, meanwhile, are the properties that influence the readability of the graph. These include minimisation of: edges crossings, the area of the graph, the number of edge bends and edge length. Other desirable aesthetics include symmetry and the maximisation of the angle between two edges incident of the same vertex (defined as the angular resolution). Optimising these properties would produce an extremely readable graph. The problem is, however, that not only is achieving these aesthetics NP-hard ([15],[16], [30]) but obtaining the optimisation of one aesthetic generally conflicts with the optimisation of another. Appendix A of [16] provides the upper and lower bounds between the aesthetics criteria for many different graphs.

The first graph drawing algorithm was given in Knuth's [49] work on visualising flow charts. Since this time there has been a plethora of work in the area and a small selection is discussed here for comparison with the proposed technique.

Fáry [26], Stein [75] and Wagner [80] independently proved that every planar graph can be drawn in the plane with straight-line edges. This result was implemented in an algorithm by Reed [59], however the resulting diagrams placed vertices very close together, thus requiring that these graphs were produced on high-resolution display devices. A similar problem was also evident in the polynomial-time algorithm of Tutte [77] which produces a straight-line drawing of a triconnected planar graph (a graph that cannot be separated by the removal of less than 3 vertices) where every interior face is a convex polygon. Chiba *et al.* [11] improved this algorithm to one that has a linear time complexity, but again, unsatisfactory layouts were produced.

The work of de Fraysseix *et al.* [14] improved upon Reed's method by presenting a technique in which planar triangulated graphs (a planar graph where every interior region has three edges) are constructed using a straight line drawing convention where every vertex is placed upon grid coordinates. The readability of these graphs is reduced, however, due to the fact that they can produce drawings with a very small angular resolution. Malitz and Papakostas [56] overcome this difficulty by showing that every planar graph of maximum degree d has minimum angular resolution of C^d radians, where C is a constant approximately equal to

0.15. Nevertheless, this is of little use for picture drawing. It is not unusual to have a high degree of discs in pictures and even for a relatively small d (in terms of pictures), 4 say, gives the minimum angle between discs as 5.1×10^{-4} radians.

A useful technique to draw a simple planar graph, Γ say, which overcomes these layout problems is given by the *visibility approach.* This method, which was first introduced in [18] and [21], involves determining a planar embedding of Γ and constructing a *visibility representation* of Γ. In a visibility representation, each vertex of Γ is represented by a horizontal line and edges are depicted by vertical line segments. Once such a representation has been constructed, the final stage in the visibility approach is to replace the horizontal lines with vertices and the vertical segments with polygonal lines. Visibility representations were introduced by Otten and van Wijk [63], who also proved that every planar graph admits such a representation. The method of Rosentiehl and Tarjan [68] constructs a visibility representation of Γ where the resulting drawing gives a more compact layout for Γ than that of Otten and van Wijk. The same method (in essence) was independently proposed by Tamassia and Tollis [76]. The method of [68], however, represents the graph by it's rotation scheme. Due to the fact that spherical pictures are represented by rotation schemes, it is the method of Rosentiehl and Tarjan [68] that is summarised here.

To construct the visibility representation of Γ, it is required that Γ is an *st-graph.*

Definition 6.1 (*st*-**graph)** *An acyclic digraph is a directed graph with has no directed cycles. A sink is vertex that contains no outgoing edges. A source is vertex with no incoming edges. An st-graph, as shown in Figure 6.1, is an acyclic digraph with a single source s and single sink t that lie on the boundary of the external region.*

A *topological numbering* can be assigned to each vertex in Γ such that for every edge (u, v) the number assigned to vertex v is greater than that of u. The *level* of the vertex v is the maximum number of paths from s to v. If Γ is not an st-graph there are various algorithms that can be implemented to transform it into an st-graph with a topological numbering ([16], Chapter 4). Rosentiehl and Tarjan [68] also assume that Γ is biconnected. If this is not the case, [83] provides linear-time algorithm to transform Γ into a biconnected graph by adding

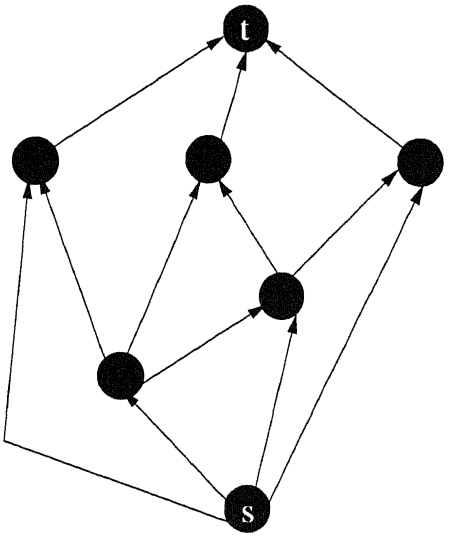


Figure 6.1: An *st*-graph

dummy edges. Once the layout is complete these edges can be ignored.

The regions of Γ are obtained by examining the rotation scheme of the graph. From the regions of Γ, the dual graph, Γ^*, can be determined. The graph Γ^* has vertex set $\mathbf{v}^*$ and edges given by $\mathbf{e}'$. For every region of Γ, there is a vertex $v^* \in \mathbf{v}^*$ that lies in its interior (with two vertices of Γ^* in the exterior region of Γ). For every edge $e \in \Gamma$ where $e \neq (s,t)$ there is an edge in Γ^*, $e^* = (f,g)$, where f lies in the region to the left of e, and g lies in the region to the right of e.

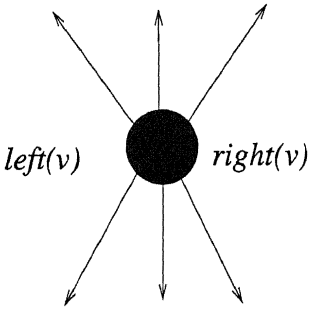


Figure 6.2: A diagram to show $left(v)$ and $right(v)$

As is proved in [76] and [83], the incoming and outgoing edges incident upon v occur consecutively around v; this is shown in Figure 6.2. The region that separates the incoming edges from the outgoing edges, in a clockwise direction around v, is denoted by $left(v)$. Similarly, $right(v)$ denotes the region around v that separates the outgoing edges from the incoming edges. The value of $left(v)$ and $right(v)$ is given by the value of the vertices of Γ^*

that lie in those regions.

A visibility representation of Γ is constructed by drawing a horizontal chain for every vertex $v \in \Gamma$. The y-coordinate of this chain is given by the level of v, and the x-coordinates are given by $left(v)$ and $right(v)$.

The x-coordinate of the vertical line segment that represents an edge, $e = (u, v)$, is given by the level of the vertex of Γ^* that lies in the region to the left of the e. The y-coordinates are given by the levels of u and v. Repeating this procedure for all vertices and edges of Γ produces the required visibility representation.

The graph drawing is then completed by replacing each horizontal segment with a vertex and each vertical line with an edge. A proof of the correctness of this algorithm is given in the paper and is based upon the correctness of the Lempel *et al.* planarity testing algorithm [51]. The drawing that has been constructed by utilising this method has a height at most $|\mathbf{v}|$ and width at most $2|\mathbf{v}| - 4$ (where $|\mathbf{v}|$ is the number of vertices in Γ). Kant [45] improved this area bound to one of $((\frac{3}{2}|\mathbf{v}|) - 3) \times (|\mathbf{v}| - 1)$; although his method requires that Γ is also triangulated.

The main problem with the visibility approach, and indeed the majority of graph drawing algorithms, is that additional constraints are placed upon the planar graph. For example, the algorithm of Rosentiehl and Tarjan [68] given above requires that Γ is a biconnected st-graph. If the graph does not satisfy these constraints, dummy edges are inserted until the desired structure for the graph is obtained. These dummy edges are then removed from the final drawing. If an aesthetically pleasing graph is produced based upon having a large number of dummy edges and these edges are then removed from it, this reduces the readability of the final drawing of the desired graph.

On the face of it the method of Rosentiehl and Tarjan [68] would be an appropriate algorithm to implement to produce drawings of pictures. It requires that a graph be represented by a rotation scheme and, from this, produces a planar, straight line drawing of the desired graph. Nevertheless the method would have to be adapted to work for non-simple graphs. Moreover, the picture may need to be made biconnected and an st-numbering would have

to be computed.

The main reason, however, why this technique cannot be used to construct drawings of pictures is based upon the fact that a picture has discs of various sizes. For instance, assume that for a picture $\mathbf{P}$, the visibility approach computed that the horizontal segment for a disc, Δ, of $\mathbf{P}$ was from 0 to 1. The disc cannot be drawn in this interval as the radius of a disc must be at least 8 units (the minimum degree of a disc is 2) and so, correspondingly, for Δ to fit on a sheet the x-coordinate of it's centre must be at least 8.

Moreover, it is not practical to assume that each interval in the visibility representation corresponds to the maximum diameter of a disc in the picture, plus a small constant to ensure adjacent discs do not touch. Take, for example, a picture over the presentation given in 1.1 (see Page 4). As is shown in Figure 1.2, this is a relatively small picture over a group presentation. The visibility representation of this picture produces a range of x-coordinates from 0 to 6. The length of the largest relator in 1.1 is 6, which gives 48 as the maximum diameter of a disc in a picture over the presentation. Hence, employing the idea that each interval in the visibility representation equals length 53 (if a value of 5 is chosen for the constant) means that the width of the sheet that the picture will be constructed on will have length $53 \times 6 = 318$ units. Now the picture in Figure 1.2 has 8 discs, only two of which have degree 6. A picture over this presentation could therefore be drawn on a much smaller sheet. For presentations with relators that consist of many letters, or whose pictures contain a number of discs, this method would produce large graphic sheets with a great deal of empty space. For this reason, it is not practical to use this method to automatically construct spherical pictures.

One way to counteract this problem is to try and remove empty areas in a large visibility representation, without altering where discs have been placed. Alternatively, the visibility representation could be employed to indicate the area of the plane where a disc, Δ say, lies. Then, by examining the location of the surrounding discs in some way, the exact location of Δ could be computed. It would be difficult to develop and implement both of these methods, however, and they imply that once a visibility representation has been constructed further

techniques are needed to transform it into a useful representation of a picture. In other words, constructing a visibility representation of a picture can only indicate where the discs should be constructed on the sheet and not provide any coordinate values for their location.

It therefore seemed practical, and more efficient, to use a different method to produce a picture representation. The Picture Visibility (PV) representation of the picture drawing algorithm involves employing a two dimensional matrix, M, to represent the plane. Each component of the matrix corresponds to an area of the plane of size $(l + 5)^2$, where l is the maximum diameter of discs in the picture. The method utilised to determine the arrangement of discs involves assigning each disc to a component of M. The position of discs in the plane can then be obtained by examining the local rotation scheme of each disc.

The matrix M therefore provides a type of visibility representation without first needing to construct a biconnected, *st*-graph of the picture, it's dual graph and an *st*-numbering of both the graph and it's dual. Moreover, if can be used to produce a representation of non-simple graphs. Admittedly, this method would not be as robust for general graphs as the visibility representation technique because it does not always place discs in the most appropriate position. This is due to the fact that two local rotations could have conflicting locations for the most appropriate place to insert the same disc into M. It is, however, thought to be a more efficient approach for constructing pictures. As shown above, the visibility representation can only suggest where discs should be placed and this information can be obtained directly from the rotation scheme of the picture.

Once the discs have been drawn on the sheets, the replacement stage is completed by constructing the arcs of $\mathbf{P}$ and drawing the basepoint on the circumference of each disc. The detailed description of the visibility and representation stages of this picture drawing algorithm is presented in the following sections.

6.2 Drawing Pictures

Given a data structure for a picture $\mathbf{P}$ over a presentation $\mathcal{P}$, the aim of this algorithm is to depict $\mathbf{P}$ graphically, with a high level of readability. The planarity and regions of a picture $\mathbf{P}$ over a presentation $\mathcal{P}$, are determined by calling `PlanarityTest`. The picture drawing algorithm comprises the following stages:

1. Construct a picture visibility representation to calculate a possible location, in the plane, for each disc.
2. Draw these discs at the required position on a picture sheet.
3. Construct the arcs of $\mathbf{P}$.
4. Once all arcs of $\mathbf{P}$ have been drawn, examine any arcs with bends. Ensure that bends are placed in a location that causes a minimum number of arc crossings in the picture drawing.
5. Store the properties of the sheet and return the drawing of $\mathbf{P}$.

6.2.1 Constructing a PV Representation

The matrix, M, that stores the PV representation is created incrementally. Firstly, the discs in the exterior region of $\mathbf{P}$, ρ_1, are inserted. As with the previous algorithms given in this thesis, ρ_1 is selected to be the region with the largest number of connection points. The technique in which the discs of ρ_1 are inserted into M depends on whether there is an odd or even number of connection points in ρ_1. The aim is to keep the arrangement of discs in M as symmetric as possible.

Let the discs in ρ_1 be given by $\{\Delta_1, \Delta_2, \cdots \Delta_n\}$. If n is even then k is said to equal to 0, otherwise $k = 1$; M is initially a $((n-k)/2+1) \times 3$ matrix. If n is even, Δ_1 is inserted at $M[1][2]$, otherwise $M[1][1] = \Delta_1$ with Δ_2 being placed at $M[1][3]$.

The remaining discs of ρ_1 are now inserted in M using the following convention:

- for $(2+k) \leq i \leq (n+k)/2$, $M[i][3] = \Delta_i$;

- for $i = (n+k)/2+1$, $M[(n+k)/2+1][2] = \Delta_i$;
- for $(n+k)/2+2 \leq i \leq n$, $M[n-i+2][1] = \Delta_i$.

This technique is illustrated in Figure 6.3.

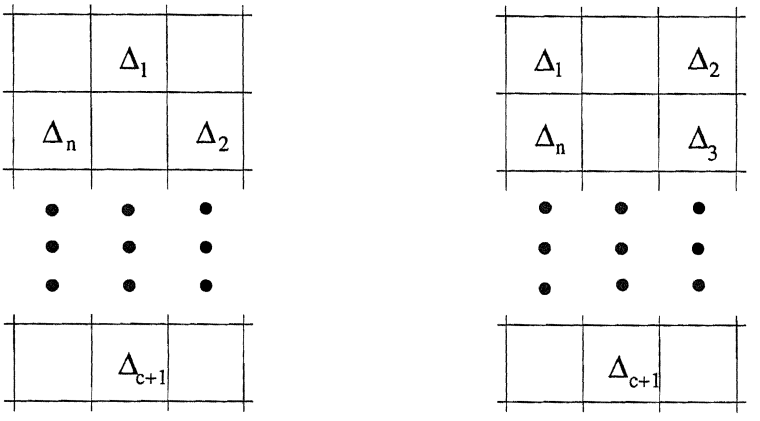


Figure 6.3: The discs of ρ_1 in M when n is even and when n is odd

The remaining discs of $\mathbf{P}$ are now inserted into M. This is achieved by examining the local rotations, firstly of discs in ρ_1, and then those of the remaining discs in M, until M contains all discs of $\mathbf{P}$. By using this method, M will always contain all the discs of $\mathbf{P}$ due to the fact that $\mathbf{P}$ is a connected graph.

Let Δ be a disc in M where the local rotation scheme for Δ is denoted by Π_Δ. The technique by which the discs in Π_Δ are inserted into M depends upon the positions, in M, of Δ and any discs of Π_Δ that are already in M. Therefore, Π_Δ is examined to find the number of discs that have been inserted into M. The following possibilities arise:

Case (1): All discs in Π_Δ have been placed into M.

Case (2): No discs in Π_Δ have been placed into M.

Case (3): One of the discs in Π_Δ have been placed into M.

Case (4): More than one, but not all, of the discs in Π_Δ have been placed into M.

If Case (1) occurs then the algorithm progresses to examine the rotation scheme of another disc in M. For Cases (2), (3) and (4) the appropriate discs are inserted into M by employing one of four methods. These methods, and the selection processes, are described below.

The Four Methods of Inserting Discs into M

Let ξ be an ordered subset of discs of Π_Δ that have not been placed into M. Essentially the discs of ξ are placed either above and below, or to the left and right of a certain point, ϑ say, in M. If either Case 2 or Case 3 arises then ϑ is given by the position of Δ in M. When Case 4 occurs, however, there are four situations where it can be more efficient to choose another position for ϑ.

Let $\{\Delta_k, \xi, \Delta_l\} \subseteq \Pi_\Delta$ where $\Delta_{k,l}$ are discs that have been inserted into M. The algorithm calculates which value to use for ϑ by examining the positions of Δ_k, Δ_l and Δ in M. The method for calculating ϑ is given in Appendix B. From this point on, for clarity, it is assumed that the position of ϑ is given by the position of Δ.

The discs of ξ are placed either vertically of horizontally around Δ. The way the discs are placed around Δ depends on which of the four types, A, B, C or D, of methods to insert discs into M have been selected. These four methods are described below and they are illustrated in Figure 6.4. Here the arrows point to the location where the discs of ξ would lie. For instance, if Type A was chosen then the disc of ξ would be placed in the row above Δ, between Δ_k and Δ_l.

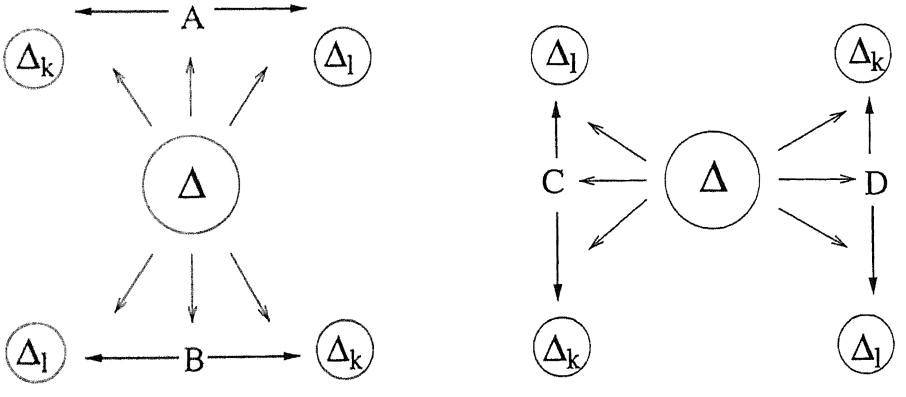


Figure 6.4: a) Illustrating types A and B b) Illustrating types C and D

Types A, B, C and D are now described in detail. Let ξ be a consecutive set of discs in Π_Δ that have not been placed into M, where $M[i][j]$ denotes the location of Δ. If the number of discs in ξ is even then $N = |\xi|/2$, if $|\xi| = 1$ then $N = 1$ otherwise $N = |\xi - 1|/2$. Furthermore, the kth disc in ξ is denoted by ξ_k (for $1 \leq k \leq |\xi|$). The four methods used to

insert discs into M are:

Type A: A new row is inserted into M above row i. If there are not N columns between Δ_k and Δ then the appropriate number of columns are added. Similarly, columns are added if there are not N columns between Δ and Δ_l. The discs of ξ are added into M as follows:

- For $1 \leq k \leq N$, the disc at ξ_k is inserted at $M[i-1][j-N+k-1]$.
- If $|\xi| \neq 1$ and $|\xi|$ is odd, then $M[i-1][j] = \xi_{N+1}$ and N is incremented by 1.
- For $N+1 \leq k \leq n$, $M[i-1][j+k-N] = \xi_k$.

Type B: A new row is inserted into M below row i. If they are required, up to N columns are added before and/or after $M[j]$. The discs of ξ are added into M as follows:

- For $1 \leq k \leq N$, the disc at ξ_k is inserted at $M[i+1][j+N-k+1]$.
- If $|\xi| \neq 1$ and $|\xi|$ is odd, then $M[i+1][j] = \xi_{N+1}$ and N is incremented by 1.
- For $N+1 \leq k \leq n$, $M[i+1][j-k+N] = \xi_k$.

Type C: A new column is inserted into M at position $j-1$. If they are required, up to N rows are added before and/or after $M[i]$. The discs of ξ are added into M as follows:

- For $1 \leq k \leq N$, the disc at ξ_k is inserted at $M[i+N-k+1][j-1]$.
- If $|\xi| \neq 1$ and $|\xi|$ is odd, then $M[i][j-1] = \xi_{N+1}$ and N is incremented by 1.
- For $N+1 \leq k \leq n$, $M[i+N-k][j-1] = \xi_k$.

Type D: A new column is inserted into M at position $j+1$. If they are required, up to N rows are added before and/or after $M[i]$. The discs of ξ are added into M as follows:

- For $1 \leq k \leq N$, the disc at ξ_k is inserted at $M[i-N+k-1][j+1]$.
- If $|\xi| \neq 1$ and $|\xi|$ is odd, then $M[i][j+1] = \xi_{N+1}$ and N is incremented by 1.
- For $N+1 \leq k \leq n$, $M[i-N+k][j+1] = \xi_k$.

Selecting which Type to Use to Insert Discs into M

For the local rotations that fall into Case (2) on Page 126, there is no advantage in selecting one type above the other three. For consistency, Type B is chosen to insert the discs of Π_Δ into M.

The way in which the relevant type is chosen for discs that fall into Case's (3) and (4) depends upon the positions, in M, of Δ and the discs that bound ξ. In Case (3) only one disc, Δ_h say, occurs in M that also occurs in Π_Δ. Therefore only the positions of Δ and Δ_h, in M, need to be considered before the appropriate method is selected. If Δ_h lies in a row above Δ then again, for consistency, Type B is selected; alternatively, Type A is chosen. Given that Case (4) arises and ξ is bounded by Δ_k and Δ_l, the positions, in M, of Δ, Δ_k and Δ_l need to be considered. It transpires, after the value of ϑ has been calculated, that there are 16 possibilities to examine for Case (4) . These possibilities are given in Appendix B.

Employing these methods to insert discs into M could alter the position, in M, of the discs in ρ_1. Once all discs have been placed in M, a procedure then ensures that the discs in ρ_1, and only these discs, lie in the first and last rows and columns of M. In other words, it is ensured that the technique used to insert the discs of ρ_1 into M, as depicted in Figure 6.3, is adhered to. The picture visibility representation for $\mathbf{P}$, M, has now been constructed.

Rows and columns are only inserted in M if there is not enough space in M to insert the discs of a local rotation. If $|\Delta|$ denotes the number of discs in $\mathbf{P}$, and n the number of discs in ρ_1, the maximum number of rows or columns to be inserted into M at any one time is given by $|\Delta| - n$. The area bound for the picture visibility representation is therefore given by $(|\Delta| - \frac{n-k}{2})^2$, where $k = 1$ if n is odd and $k = 0$ otherwise. This compares favourably with the area bound of the methods of Rosentiehl and Tarjan, which is $|\Delta| \times (2|\Delta| - 4)$ [68], Kant $((\frac{3}{2}|\Delta|) - 3) \times (|\Delta| - 1)$ [45] and the orthogonal grid drawing algorithm of Biedl and Kant, $|\Delta|^2$ [2].

Creating the PV representation of $\mathbf{P}$ in such a way minimises the number of arc crossings. This is achieved in two ways, firstly, by using the rotation scheme of $\mathbf{P}$ to insert discs into M. Moreover, the technique of placing a subset, ξ, of discs in Π_Δ, say, as close to Δ as possible

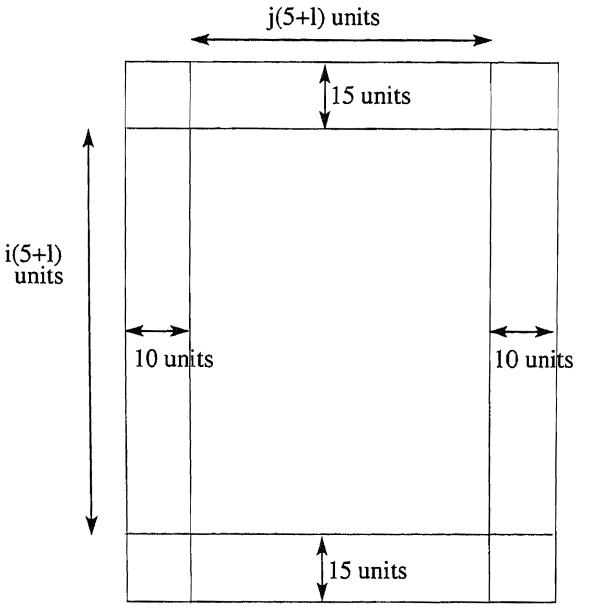


Figure 6.5: The template for a picture sheet

(i.e. in the preceding or following row or column) also limits the amount of arc crossings. Arc crossings do still occur, however, due to the fact that the most suitable location for a disc, as dictated by one local rotation scheme, may not necessary be the same position as that dictated by another local rotation scheme.

Once M has been created, the picture sheet for $\mathbf{P}$ can be constructed. Let L be the length of the largest relator of $\mathcal{P}$. The diameter of the largest disc must therefore be given by $l = 4 * L * 2$. Each component of M therefore represents $(5 + l)^2$ units of the plane, where the remaining 5 units ensures that if two of the largest discs are adjacent in M, their circumferences do not touch on the picture sheet.

If M has i rows and j columns, then the picture sheet has height $(i * (5 + l)) + 30$ and width $(j * (5 + l)) + 20$. The constants 30 and 20 provide a small border around the picture drawing which enables the labels of arcs in the exterior region to be clearly viewed. Figure 6.5 gives the template for a picture sheet which has been created from M.

The picture sheet, for a picture over $\mathcal{P}$, is created with `PictureEditor` which takes the name of the sheet, it's width, height and $\mathcal{P}$ as input. The `PictureDisc` function is then used to construct discs onto the sheet. Recall that `PictureDisc` requires the coordinates for the centre of the disc and a label as input. Coordinates can be determined from a disc's

position in M and the label of every disc is stored in the relator information of $\mathbf{P}$. In this way every disc in $\mathbf{P}$ can be drawn onto the sheet in the position dictated by M.

6.2.2 Drawing Arcs of Pictures

The next stage of the algorithm is to draw the arcs of $\mathbf{P}$ on the picture sheet. The arcs of the exterior region are drawn first. These arcs are constructed in a different way to the arcs of the interior region of $\mathbf{P}$; the reason for this is to ensure that the area for the interior regions of $\mathbf{P}$ is maximised whilst maintaining readability.

If there are only two discs in ρ_1, the corresponding arcs are constructed using two vertical lines between the two smallest and two largest x coordinates of the circumferences of the two discs. For pictures where the number of discs is greater than two, the drawing of $\mathbf{P}$ adopts a rectangular shape, as shown in Figure 6.6.

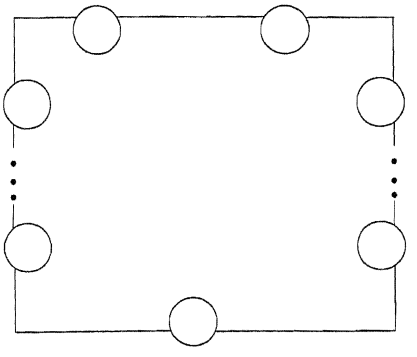


Figure 6.6: A diagram to illustrate the rectangular shape of $\mathbf{P}$

Drawing the remaining arcs of $\mathbf{P}$ is more complex. These arcs are drawn using the `Arc` and `ArcWithBends` functions. The `Arc` function constructs arcs with at most 3 lines. If an arc has more than 3 lines, the coordinates for the endpoints of these additional lines are computed by referring to the PV representation for $\mathbf{P}$: if there are any discs in M that lie on the route that an arc takes, a 'bend' in the arc is constructed at an appropriate point.

The arcs of each interior region are constructed in turn. Let α be an arc of $\mathbf{P}$ that is incident on two discs, Δ_1 and Δ_2 say, where (x_{c_k}, y_{c_k}), for $k = \{1, 2\}$, denotes the coordinates of the centre of Δ_k. The technique adopted to construct α is summarised in the following stages:

1. The endpoints of α are computed using the `Arc` function.

2. The PV representation of $\mathbf{P}$ is examined to determine if α intersects any other discs of $\mathbf{P}$. If this is the case, it is circumvented by constructing appropriate bends in α, and the arc is drawn using the `ArcWithBends` function.

3. Once all arcs of $\mathbf{P}$ have been constructed, a 'tidying-up' process begins. If bends were placed in α in Stage (2), the location of these bends are investigated to ensure that drawing each bend at the given point does not cause any arc crossings. If this is the case, a procedure attempts to find a new location for this bend such that any arc crossings are avoided. If no such point can be found, however, that bend is deleted and that line of α is re-drawn such that it intersects that disc.

Stage (2) and (3) will now be described in more detail. Given that Stage (1) has been implemented, the path that α takes on the picture drawing sheet is now known. The visibility representation of $\mathbf{P}$, given by M, is utilised to determine if α intersects any discs (other than $\Delta_{1,2}$). Assume that Δ_1 lies at $M[i_1][j_1]$ and Δ_2 lies at $M[i_2][j_2]$. If any of the components of M from $M[i_1][j_1]$ to $M[i_2][j_2]$ contain a disc of $\mathbf{P}$, then α must intersect this disc. This disc is added to a list, $\mathcal{L}$ say, such that when all components of M between $M[i_1][j_1]$ and $M[i_2][j_2]$ have been examined, $\mathcal{L}$ contains all the discs that α would intersect.

Let Δ be a disc in $\mathcal{L}$ with centre point (x, y) and radius r. If $i_1 \neq i_2$ or $j_1 \neq j_2$, the coordinate point of the bend of α needed to avoid this intersection is placed in one of four possible locations. If $i_1 = i_2$ or $j_1 = j_2$ only two locations need to be examined. These locations are shown in Figure 6.7. On this diagram, $x_{1,2}$ and $y_{1,2}$ refer to the coordinates of the line of α that intersects Δ. Initially the bend is always placed at position the B_1.

This method is repeated for all discs in $\mathcal{L}$ and, consequently, the coordinates for the endpoints of the lines of an α are now known. The arc can then be drawn using either the `ArcWithBends` or `Arc` functions. Note that the remaining input values required for these functions: the discs, $\Delta_{1,2}$, arc numbers and label of α, can be obtained directly from the way that α is expressed in the region.

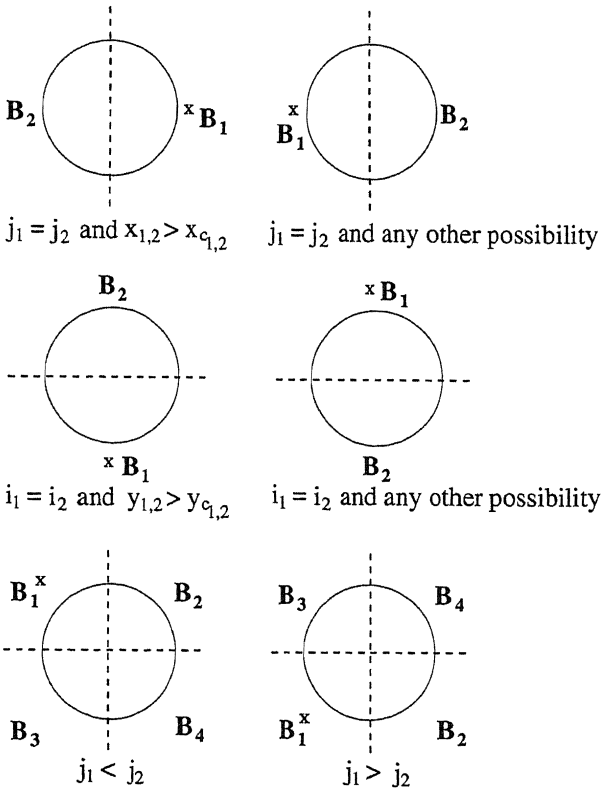


Figure 6.7: A drawing to illustrate the possible locations of the bend

If $\mathcal{L}$ is not empty, it is placed together with the properties of α into another list, `CheckArcs`. Once all arcs of $\mathbf{P}$ have been constructed, `CheckArcs` then contains a list of arcs which need to be inspected to ensure that their bends have been put in the most appropriate place on the picture sheet.

Let α contain an initial bend near the boundary of Δ. The number of arcs that are incident in the section containing B_1 of $\delta\Delta$ are counted. If there are no arcs emanating from Δ in this section the next arc in `CheckArcs` is examined. If, however, there are arcs emanating from this section, the point where the bend lies needs to be moved, as it clearly causes arc crossings. The number of arcs of Δ whose connection points emanate from the remaining sections of Δ are then examined in order, beginning from the section that contains B_2. If no connection points of Δ emanate from a particular section of Δ, the point of the bend is constructed here, using the `MoveArcLine` function.

If the situation arises that arcs are incident on Δ in all areas around the disc then the bend is deleted and α is re-drawn so that it intersects the disc. It is true that this option

does reduce the readability of the picture. Nevertheless the resulting picture drawing is still clearer than if a bend was placed in the area containing the least arc endpoints. Furthermore, as Figure 6.8 depicts, there is the possibility that placing the bend in a section with no arc endpoints could still cause arc crossings. Clearly there is no easy way to ensure that an arc crossing does not occur, but it is felt that the method given here is the most efficient technique to use.

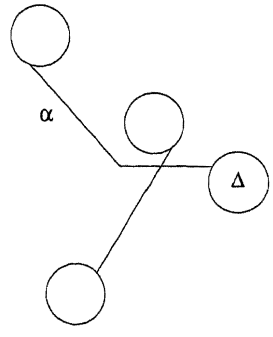


Figure 6.8: A drawing to illustrate how arc crossings can still occur

In conclusion therefore, it is believed that this an efficient method to ensure there is a minimum number of arc and discs crossings in the resultant picture. Once all the bends of the picture have been examined, and if necessary re-drawn, the `Basepoint` procedure constructs basepoints on each disc on the sheet and the picture drawing is complete. The final stage consists of saving the properties of the sheet. The data structure of the picture, the group presentation that the picture represents, the discs and arcs that have been created on the sheet and, if required, the number of the picture in the generating set of a presentation, are stored.

6.3 Implementation, Experiments and Conclusions

The above algorithm was implemented in a function, `DrawPicture`, and takes as input a presentation and a data structure of a picture, or sets of pictures, over that presentation. The data structures for the generating pictures of the 45 presentations given in Appendix C were obtained by using the `PictureFromIdentitySequence` function. `DrawPicture` was then tested on each of these 45 sets. The resulting picture drawing fell into one of three

categories: drawings that needed no further user-manipulation, those that required a small amount of user-manipulation (less than 5 alterations) and, thirdly, picture drawings that needed more than 5 manipulations.

The results of this experiment are shown in Table 6.1, where m denotes the number of manipulations. There are 158 pictures in these 45 sets with 81 of these pictures being complete dipoles. Complete dipoles created using this picture drawing algorithm do not require any further user-manipulation. Due to the fact that they occur so frequently in these 45 sets they were removed from the results given in Table 6.1, so as not to bias the overall result.

Manipulations required	Number of pictures
$m = 0$	27
$1 \leq m \leq 5$	40
$m > 5$	10

Table 6.1: Results of the picture drawing algorithm

A selection of drawings that this algorithm produces are given in the examples below.

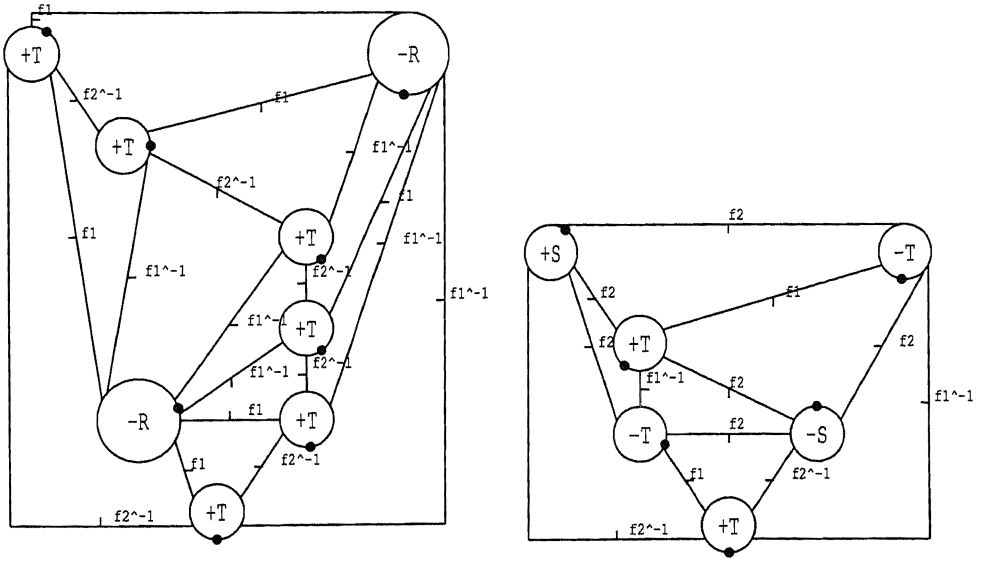


Figure 6.9: Generating pictures over a presentation for C_2Q_3

Example 6.2.1

Figure 6.9 shows two of the generating pictures for π_2, for a presentation over the group C_2Q_3, $< f1, f2; f1^6, f2^4, f2^{-1}f1f2f1 >$. The relators are labelled by R, S and T respectively. As can be seen, these diagrams that the picture drawing algorithm has produced are

aesthetically pleasing and therefore require no further manipulation.

Example 6.2.2

A generating picture for a presentation over the group A_4, $< f1, f2; f1^2, f2^2, (f1f2)^3 >$ is depicted in Figure 6.10. Again the relators are labelled by R, S and T respectively. This diagram does require a few alterations as there are several arc crossings and some confusion as to which arc connects to which disc in the right side of the drawing. The modified drawing of Figure 6.10 is shown in Figure 6.11.

Example 6.2.3

A third example of the drawings produced using this technique is given in Figure 6.12. This drawing is a generating picture for a presentation, $< f1, f2; f1^2, f2^9, (f1f2)^2 >$, over the group D_9 and does require some amount of manipulation to produce a readable picture. Again, the relators are labelled by R, S and T respectively. The modified drawing of Figure 6.12 is given in Figure 6.13. Note that the sizes of Figures 6.12 and 6.13 have been reduced to fit onto an A4 page. For the three above examples, the remaining generating pictures over π_2 are dipoles and are not drawn as they require no further manipulation.

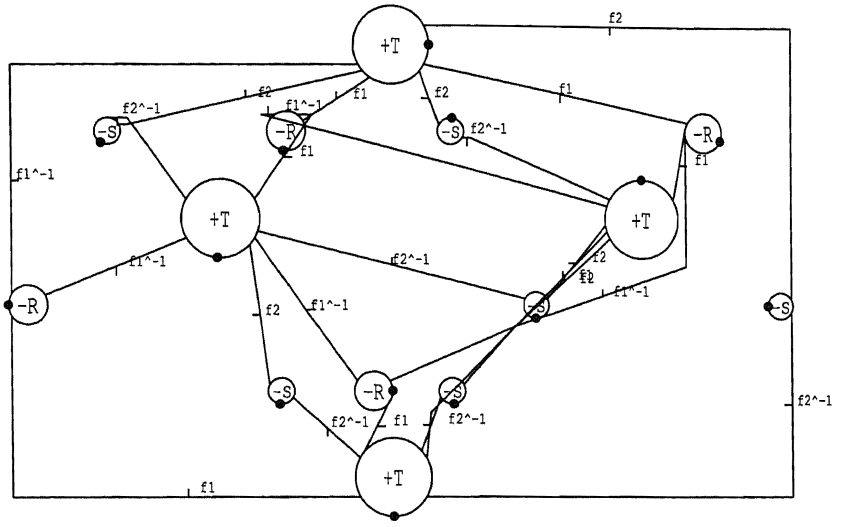


Figure 6.10: A generating picture over a presentation for A_4

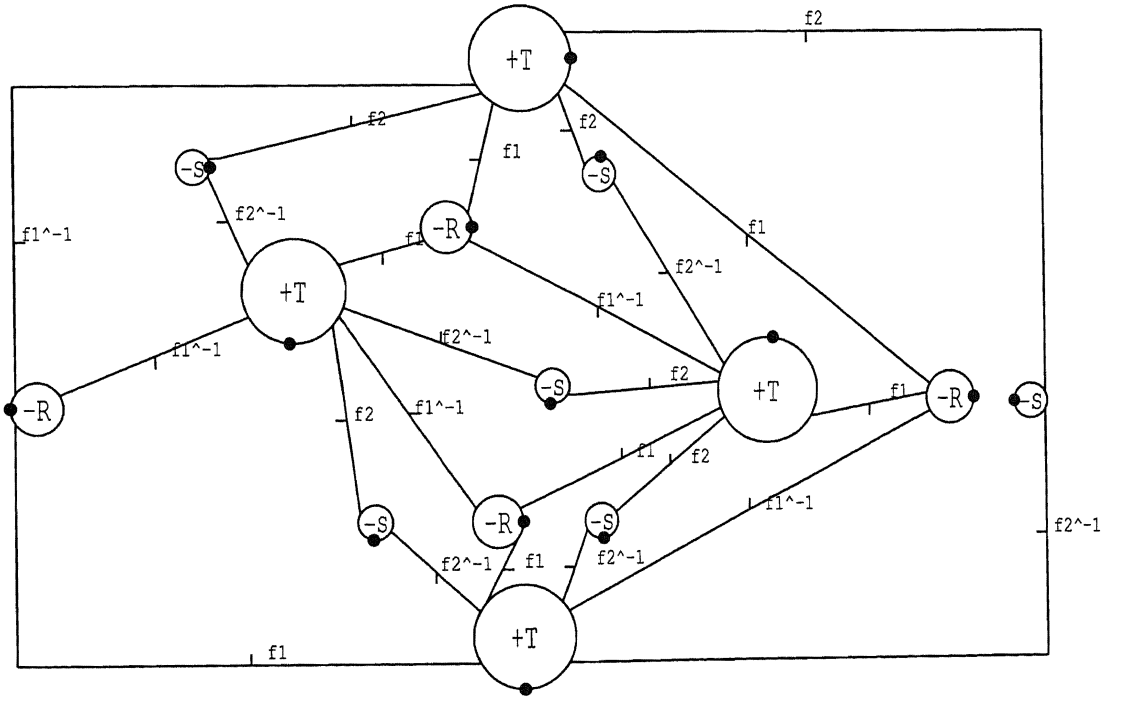


Figure 6.11: A generating picture over a presentation $< f1, f2; f1^2, f2^2, (f1f2)^3 >$

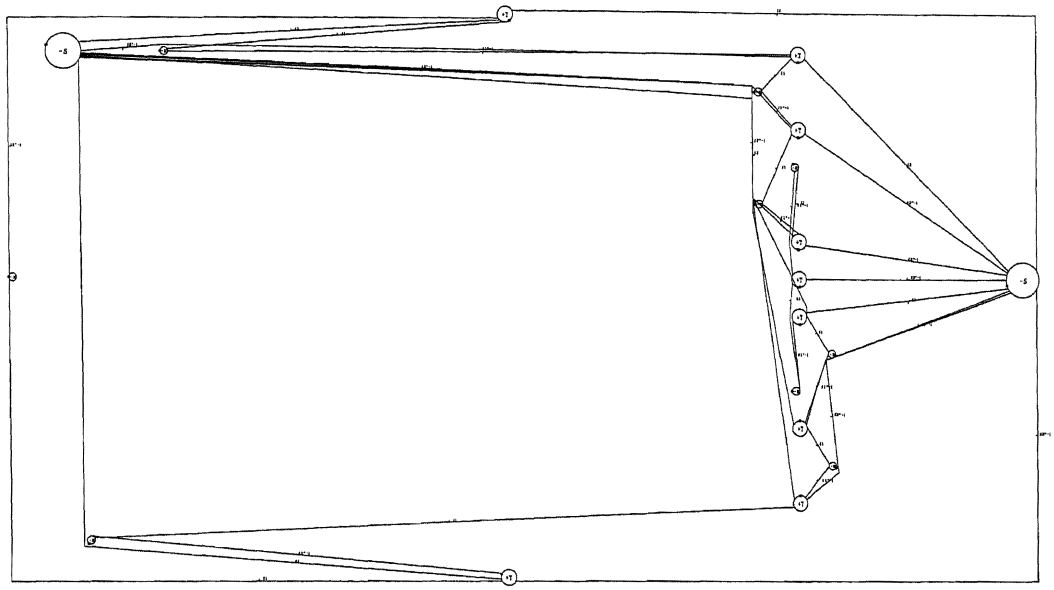


Figure 6.12: A generating picture over a presentation for D_9

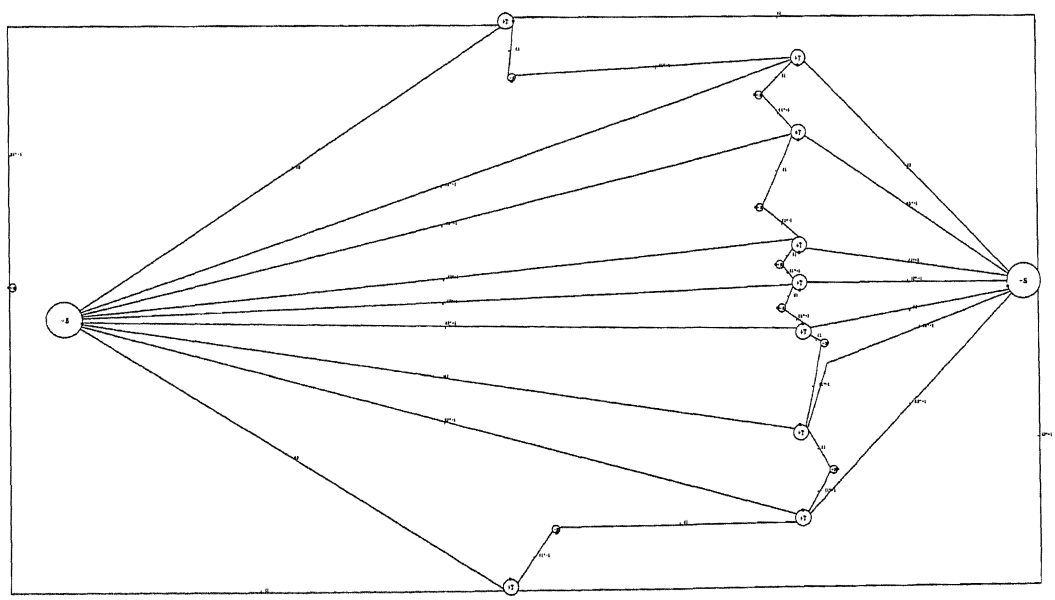


Figure 6.13: A generating picture over a presentation $< f1, f2; f1^2, f2^9, (f1f2)^2 >$

The algorithm presented above does not always create perfect pictures; as can be seen from Table 6.1 it is not unusual for a few user manipulations to be performed to produce an aesthetically pleasing drawing. Arc crossings still occur due to the fact that discs could have been placed in a more appropriate position on the sheet, or that the coordinates for the endpoints of arcs could have been placed in a more suitable location. One way to avoid arc crossings would be take each line of each arc in the picture and by using the equations of the lines of the remaining arcs, determine if an arc crossing occurs. The problem with this polynomial method is that when an arc crossing is discovered, there is no simple procedure to change the coordinates of one of the lines endpoints to avoid such a crossing. Indeed if a line was redrawn, all the lines in the picture would have to re-examined to ensure that redrawing the line in a new position does not cause a further crossing.

A further technique which would circumvent arc crossing would be to create the regions of $\mathbf{P}$ in turn, as in the picture drawing algorithm, but to also store the area that the region has been constructed in. If an arc, α say, in a subsequent region then crosses this into this area, the resulting arc crossing can be avoided by drawing bends of α outside this area. It would, however, be quite a complicated procedure to store the areas covered by a region. The endpoints of each line of each arc of the region could be saved but its is difficult to construct the area of a region from these endpoints. The PV representation cannot be used in this situation as each component of the corresponding matrix represents a section of the plane, and not individual coordinate points. Moreover, even if a matrix could be used to depict every point in the plane there is still no simple way to store the area covered by an interior region of a picture.

Theoretically, this could be achieved by creating such a matrix for every interior region of $\mathbf{P}$, where every component equals 0. Once an arc of a region, ρ_k say, has been drawn, every point of the plane where the arc has been embedded can be transformed into a value of 1 in the corresponding matrix, M_k say, for ρ_k. This method can then be repeated for all arcs of ρ_k.

Each row of M_k can then be examined. If a section of 0 entries in a row are bounded by

components of value 1, these 0 entries must represent points of the plane that are interior to ρ_k. These 0 components can then be replaced by those of value 1. In this way the area of a plane that a region of a picture represents can be stored. When constructing subsequent arcs of the a picture, each matrix can be examined to ensure that the endpoints of the lines of an arc do not occur in the interior of an existing region. Once all arcs have been constructed, all the matrices of the interior regions of the picture can be investigated to ensure that no two regions occupy the same subsection of the plane. If this is the case, the arcs of one region can be moved and checked until a complete planar picture drawing is produced. This latter stage would be quite a complicated procedure to implement. Clearly this method requires a great deal of storage space, even for relatively small pictures. Furthermore by iteratively comparing such large matrices gives this algorithm an exponential time complexity. For these reasons it was decided to use the picture drawing method that has been implemented here and leave these theoretical technique as a potential subject for further work.

This picture drawing algorithm produces pictures that may need some manual manipulation to improve readability. These manipulation functions are described in Chapter 5 and are easy for a user to execute. The rule checking procedures that have been encoded into these functions ensure that only a "valid" picture over a presentation is produced. For instance, the endpoint of an arc cannot be moved to another location on a disc's boundary if drawing it there would contravene the local rotation scheme of that disc. Given that it is easy to manually modify a picture, it is felt that algorithm, combined with the work in Chapter 5, provide a useful way to produce a visualisation of a spherical picture.

Chapter 7

Conclusions and Further Work

The work presented in this thesis describes algorithms and software pertaining to the second homotopy module, π_2, of group presentations. The primary aim of this work was to develop a tool that would assist with the study of π_2, which in turn would provide a deeper understanding of this homotopy module. SPICE contains all the algorithms presented in this thesis and it is hoped that future geometric and algebraic algorithms pertaining to π_2 will be incorporated into this package. For example, work is currently being undertaken by the author, Carsten Maple and Steve Pride on implementing the algorithm of Cruickshank and Pride [13]. This work uses the geometry of spherical pictures to compute a particular group invariant, namely higher dimensional Alexander ideals.

The algorithms given here are essentially based upon the properties of spherical pictures, although the fact that Chapter 3 provides methods to translate between spherical pictures and identity sequences implies that π_2 elements can be entered either geometrically or algebraically. These two functions are therefore extremely useful techniques, both for the work presented here and for future algorithms pertaining to π_2.

A geometric algorithm is presented in Chapter 4. Here an algorithm has been implemented that determines if a given presentation represents a group extension. This work could be extended by implementing Theorem 6.4 of [1], which computes a 3-presentation of a group extension. Once a 3-presentation of a group, Q say, has been determined the admissible

presentation functions can then, in turn, be used to find a group presentation for the extension of an arbitrary group K by Q. Again, a 3-presentation could then be found for this group extension. Using this method to compute 3-presentations provides a minimal set of generates for π_2. Once such a set of π_2 generators for a presentation is found, it can be added to the SPICE database.

The computational packages of Ellis and Kholodna ([24] and Heyworth and Wensley [34] also provide methods to produce minimal generating sets of π_2 (although note that in this later work the results in the sequel paper [33] are required). These two methods could be used to obtain a minimal generating set of identity sequences and the corresponding geometric description of these identity sequences can be obtained from the `PictureFromIdentitySequence` function. Once more, these pictures can be added to the database. The database is therefore a library of π_2 modules that can be used for reference purposes, and it additionally provides data to run experiments on future algorithms pertaining to π_2.

The algorithms of Chapter 3 could provide a method to automatically generate π_2 elements. There are a variety of techniques that have been employed which automatically produce graphs; [17] uses such a procedure to test algorithms upon and the GDToolkit [32] is a software package that automatically draws graphs according to a desired aesthetic criteria. The techniques used by these programs could be adapted and combined with the work here to automatically produce a picture over a presentation. It is thought that this would involve specifying a maximum number of discs to be used for each relator and implementing a searching algorithm to "connect" corresponding connection points emanating from each disc. If a picture over a presentation could be produced, the `IdentitySequenceFromPicture` function could be used to determine if this picture depicted an identity sequence for that presentation.

A useful way to encode the geometrical and combinatorial content of spherical pictures is presented in Chapter 2, and is based upon the idea of rotation schemes. For the purposes of spherical pictures, however, the traditional use of rotation schemes has been modified. Firstly, rotation schemes are used here to represent both planar and non-planar graphs;

previously rotation schemes have only been assigned to planar graphs. This requirement is necessary for situations where typographical errors mean that a given rotation scheme does not represent a spherical picture. Secondly, the labels of arcs in pictures can represent both a generator and it's inverse; this is opposed to standard graphs where an edge has only one label. This problem is overcome with the convention that the label of each connection point is stored, with respect to a clockwise orientation of the disc it is incident upon.

The certainty that a given data structure represents a spherical picture is verified with the planarity testing algorithm and the function `CheckDiscs`, as described in Chapter 2. If the data structure does depict a planar graph the regions of the picture are produced. Obtaining the regions of a picture is an essential requirement for the technique by which an identity sequence is obtained from a representation of a picture (Chapter 3) and the picture drawing algorithm (Chapter 6).

The picture drawing algorithm given in Chapter 6 is an initial attempt to graphically depict a picture from its data structure. The resulting drawing constructed by this technique may have some arc crossings and therefore manual modifications are required to produce an aesthetically pleasing and readable picture drawing. These manipulation functions are easy to use and have "rule checking" procedures which ensure that a user cannot change a picture into an invalid picture. This picture drawing algorithm could be improved upon in a number of ways, as suggested in Section 6.3. These improvements, however, would increase the computational time of the algorithm.

A picture can also be drawn manually on a SPICE picture sheet. Once a picture has been drawn, the function `EnterPicture` computes it's corresponding data structure. Obviously if there is a drawing error with the picture this is signalled by `EnterPicture`. Pictures are constructed with the picture disc, arcs and basepoints graphic objects, which have been created from existing XGAP graphic objects. The arc picture object could be improved in a number of ways. Ideally a function could be implemented where splines are used for arcs as opposed to a collection of straight lines. This idea was investigated but it proved to go beyond both the capabilities of XGAP and the time constraints of this project. Furthermore, by

developing the fundamental properties of XGAP more shortcut procedures could be encoded so that a user could draw an arc directly onto a picture sheet, as with existing drawing packages. Increasing the number of shortcuts would also help the user when utilising the `MoveArcLine` function. As with the other shortcut procedures, the required line could be selected and dragged to the desired place on the picture sheet.

Both the planarity testing algorithm and the picture drawing technique also have applications in other disciplines. One of the main advantage of these methods over existing methods is that they are valid for non-simple graphs. Most previous research into planarity testing and graph drawing focuses on simple graphs only. The planarity testing algorithm can be used on arbitrary graphs in areas such as genetics, communication design, VLSI and network optimisation; disciplines where the order of the arcs incident on discs could be important in the representation of the system.

The picture drawing function provides a way to draw graphs where it is useful to have a variety of vertex sizes, or where vertices represent different objects. For example, in the modelling of a sludge wastewater treatment plant (see [81] for an example) there are a variety of node (vertex) types that are required to represent features of the system: aeration tank, sedimentation tank and pumps, for example. Separate labels and sizes could be assigned to each type of node and the picture drawing algorithm could be applied to produce a layout of such a system.

Furthermore, in order to graphically produce spherical pictures, the concept of a directed edge had to be created. This graphic object can now be used to create graphs of groups, which is a request that has been made on the GAP forum (see [62]).

In conclusion the work given here provides a basis for an XGAP share package for investigating π_2 modules over group presentations. It has given rise to several areas for further work, from the automatic generation of π_2 elements to implementing other geometric algorithms. Moreover, several of the applications can be used independently in different disciplines. It is felt that the aim to introduce a tool to study the second homotopy module of a presentation has been fulfilled.

Bibliography

[1] Baik, Y.G., Harlender, J., Pride, S.J., (1998) The Geometry of Group Extensions, *Journal of Group Theory*, **1**, 395-416.

[2] Biedl, T., Kant, G., (1998) A Better Heuristic for Orthogonal Graph Drawings, *Computational Geometry: Theory and Applications*, **9**, 159-180..

[3] Bogley, W.A., Pride, S.J., (1992) Aspherical Relative Presentations, *Proceedings of the Edinburgh Mathematical Society*, **35**, 1-39.

[4] Bogley, W.A., Pride, S.J., (1993) Calculating Generators of π_2. In C.Hog-Angeloni, W.Metzler, A.Sieradski (eds.) *Two-Dimensional Homotopy and Combinatorial Group Theory*, LMS Lecture Note Series 197, New York: Cambridge University Press, 157-188.

[5] Booth, K.S., Lueker, G.S., (1976) Testing the Consecutive Ones Property, Interval Graphs, and Graph Planarity using PQ-tree Algorithms, *Journal Computation Systems*, **13**, 335-379.

[6] Bridgeman, S.S., Di Battista, G., Didimo, W., Lotta, G., Tamassia, R., Vismara, L., (1999) Turn Regularity and Planar Orthogonal Drawings, *Lecture Notes in Computer Science*, **1731**, 8-26.

[7] Brown, R., Huebschmann, J., (1979) Identities Among Relations. In R.Brown, T.L.Thickstun (eds.) *Low Dimensional Topology: Proceedings of Conference on Topology in Low-Dimension, Bangor, 1979*, London: Cambridge University Press, 153-201.

[8] Brown, R., Salleh, A.R., (1998) Free Crossed Resolutions of Groups and Presentations of Modules of Identities Among Relations, *LMS Journal of Computation and Mathematics*, **2**, 28-61.

[9] Celler, F., Neunhöffer, (1999) *The XGAP Manual*, (`http://www.gap-system.org/share/xgap.html`)

[10] Chiba, N., Nishizeki, I., Abe, S., Ozawa, T., (1985) A Linear Algorithm for Embedding Planar Graphs Using P-Q Trees, *Journal of Computer Systems and Science*, **30**, 1, 54-76.

[11] Chiba, N., Yamanouchi, T., Nishizeki, T., (1984) Linear Algorithms for Convex Drawings of Planar Graphs. In J.A.Bondy, U.S.R.Murty (eds.) *Progress in Graph Theory*, New York: Academic Press, 153-173.

[12] Collins. D.J, Huebschmann, J., (1982) Spherical Diagrams and Identities Among Relations, *Annals of Mathematics*, **261**, 155-183.

[13] Cruickshank, D.A., Pride, S.J., (2001) *New Invariants for Groups*, Preprint, University of Glasgow.

[14] de Fraysseix, H., Pach, J., Pollack, R., (1990) How to Draw a Planar Graph on a Grid, *Combinatorica*, **10**, 41-51.

[15] Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G., (1994) Algorithms for Drawing Graphs: an Annotated Bibliography, *Computational Geometry: Theory and Applications*, **4**, 5, 235-282.

[16] Di Battista, G., Eades, P., Tamassia, R., Tollis, I.G., (1999), *Graph Drawing*, New Jersey: Prentice Hall.

[17] Di Battista, G., Garg, A., Liotta, G., Tamassia, R., Tassinari, E., Vargiu, F., (1997) An Experimental Comparison of Four Graph Drawing Algorithms, *Computational Geometry*, **7**, 303-325.

[18] Di Battista, G., Tamassia, R., (1988) Algorithms for Plane Representations of Acyclic Graphs, *Journal of Theoretical Computer Science*, **61**, 175-198.

[19] Di Battista, G., Tamassia, R., (1989) Incremental Planarity Testing, *Proceedings of the 30th Annual IEEE Symposium on the Foundations of Computer Science*, 436-441.

[20] Di Battista, G., Tamassia, R., (1996) On-Line Planarity Testing, *Siam Journal of Computation*, **25**, 5, 956-997.

[21] Di Battista, G., Tamassia, R., Tollis, I., (1992) Constrained Visibility Representations of Graphs, *Information Processing Letters*, **41**, 1-7.

[22] Duncan, A., Howie, J., (1992) Weibaum's Conjecture on Unique Subwords of Nonperiodic Words, *Proceedings of the American Mathematical Society*, **115**, 4, 947-955.

[23] Edjvet M., Thomas, R., (1997) The Groups $(l, m|n, k)$, *Journal of Pure and Applied Algebra*, **114**, 175-208.

[24] Ellis, G., Kholodna, I., (1999) Three-Dimensional Presentations for the Groups of Order at most 30, *LMS Journal of Computation and Mathematics*, **2**, 118-139.

[25] Fan, K., Chang, C., (1991) Surface Extraction from Line Drawings of a Polyhedron, *Pattern Recognition Letters*, **12**, 627-633.

[26] Fáry, I., (1948) On Straight Line Representations of Graphs, *Acta Scientiarum Mathematicarum*, **11**, 229-233.

[27] Fenn, R.A., (1983) *Techniques of Geometric Topology*, LMS Lecture Note Series 57, London: Cambridge University Press.

[28] The GAP-Group, (2000) *GAP – Groups, Algorithms and Programming, Version 4.2*, Aachen, St-Andrews, (`http://www.gap-system.org`)

[29] The GAP-Group, (2000) *The GAP 4 Manual*, (`http://www.gap-system.org/Info4/manual.html`)

[30] Garg, A., (1995) On Drawing Angles in Graphs, In R.Tamassia, I.Tollis (eds.) *Graph Drawing (Proc. GD 1994), Volume 894, Lecture Notes of Computer Science*, Springer-Verlag, 84-95.

[31] Garey, M.R., Johnson, D.S., (1983) Crossing Number is NP-Complete, *SIAM Journal of Algebraic Discrete Methods*, **4**, 3, 312-316.

[32] The GDToolkit Homepage, (`http://www.dia.uniroma3.it/~gdt/index.html`)

[33] Heyworth, A., Reinert, B., (1999) *Reduction of Generating Sets of* $\mathbf{Z}G$*-modules and Applications to Identities Among Relations*, Preprint, University of Bangor.

[34] Heyworth, A., Wensley, C.D., (1999) *Logged Rewriting Procedures with Application to Identities Among Relations*, Preprint, University of Bangor.

[35] Hitchman, M.P., (2000) An Identity Property for 2-Complex Pairs, *Glasgow Journal of Mathematics*, **42**, 299-318.

[36] Hog-Angeloni, C., Metzler, W., (1993) Geometric Aspects of Two-Dimensional Complexes. In C.Hog-Angeloni, W.Metzler, A.Sieradski (eds.) *Two-Dimensional Homotopy and Combinatorial Group Theory*, LMS Lecture Note Series 197, New York: Cambridge University Press, 1-50.

[37] Hopcroft, J., Tarjan, R., (1974) Efficient Planarity Testing, *Journal of the Association for Computing Machinery*, **21**, 4, 549-568.

[38] Hope, A.K., (1971) A Planar Graph Drawing Algorithm, *Software Practice and Experience*, **1**, 83-91.

[39] Howie, J., (1989) The Quotient of a Free Product of Groups by a Single High-Powered Relator. I. Pictures. Fifth and Higher Powers. *Proceedings of the London Mathematical Society*, **3**, 59, 507-540.

[40] Howie, J., (1990) The Quotient of a Free Product of Groups by a Single High-Powered Relator. II. Fourth Powers. *Proceedings of the London Mathematical Society*, **3**, 61, 33-62.

[41] Howie, J., Metaftsis, V., (2001) On the Asphericity of Length Five Relative Group Presentations, *Proceedings of the London Mathematical Society*, **82**, 173-194.

[42] Jakoby, A., Reischuk, R., Schindelhauer, C., (1998) The Complexity of Broadcasting in Planar and Decomposable Graphs, *Discrete Applied Mathematics*, **83**, 1-3, 179-206.

[43] Jiang, X.Y., Bunke, H., (1993) An Optimal Algorithm for Extracting the Regions of a Plane Graph, *Pattern Recognition Letters*, **14**, 553-558.

[44] Johnson, D.L., (1997) *Presentations of Groups*, 2nd Edition, Cambridge: University Press.

[45] Kant, G., (1997), A More Compact Visibility Representation, *International Journal of Computational Geometry and Applications*, **7**, 3, 197-210.

[46] Kilgour, C., (1995) *Using Pictures in Semigroup Theory*, PhD Thesis, University of Glasgow.

[47] Kilgour, C., Pride, S.J., (1996) Cockcroft Presentations, *Journal of Pure and Applied Algebra*, **106**, 275-295.

[48] Kilibarda, V., (1996) On the Algebra of Semigroup Diagrams, *International Journal of Algebra and Computation*, **7**, 313-338.

[49] Knuth, D., (1963) Computer Drawn Flowcharts, *Communications of ACM*, **6**, 9, 555-563.

[50] Kuratowski, C., (1930) Sur le Probleme des Corbes Gauches en Topologie, *Fundamenta Mathematicae*, 271-283.

[51] Lempel, A., Even, S., Cederbaum, I., (1967) An Algorithm for Planarity Testing of Graphs. In P.Rosenstiehl (ed.) *Theory of Graphs*, New York: Gordon and Breach, 215-232.

[52] Liu, Y., Morgana, A., Simeone, B., (1998) A Linear Approximation for 2-Bend Embeddings of Planar Graphs in the 2D Grid, *Discrete Applied Mathematics*, **81**, 1-3, 69-91.

[53] Lyndon, R.C., Schupp, P.E., (1977) *Combinatorial Group Theory*, New York: Springer.

[54] The Magma Computer Algebra Homepage,
(`http://www.maths.usyd.edu.au:8000/magma`)

[55] Magnus, W., Karrass, A., Solitar, D., (1966) *Combinatorial Group Theory*, New York: Wiley.

[56] Malitz, S., Papakostas, A., (1992) On the Angular Resolution of Planar Graphs, *Proceedings of the 24th Symposium on the Theory of Computing*, 527-538.

[57] Mehlhorn, K., Mutzel, P., (1996) On the Embedding Phase of the Hopcroft and Tarjan Planarity Testing Algorithm, *Algorithmica*, **16**, 233-242.

[58] Mehlhorn, K., Näher, S., (1995) *A Platform for Combinatorial and Geometric Computing*, Cambridge: Cambridge University Press.

[59] Mcdiarmid, C., Reed, B., Schrikver, A., Shepard, B., (1994) Induced Circuits in Planar Graphs, *Journal of Combinatorial Theory, Series B*, **60**, 2, 169-176.

[60] Mohar, B., (1999) A Linear Time Algorithm for Embedding Graphs in an Arbitrary Surface, *SIAM Journal of Discrete Mathematics*, **12**, 6-26.

[61] Mondshein, L., (1971) *Combinatorial Orderings and the Geometric Embeddings of Graphs*, Technical Note 1971-35, Lincoln Laboratory, Massachusetts Institute of Technology.

[62] Neubueser, J., (2000) *Re: Graph of a Group (Group Diagram)?*, email to GAP forum, dated 18th December 2000.

[63] Otten, R.H.J.M., van Wijk, J.G., (1978) Graph Representations in Interactive Layout Design, *Proceedings of the IEEE International Symposium on Circuits and Systems*, 914-918.

[64] Pride, S.J., (1990) Identities Among Relations of Group Presentations. In E.Ghys, A.Haeflinger, A.Verjousky (eds.) *Group Theory from a Geometrical Viewpoint*, Trieste 1990, New York: World Scientific Publishing, 686-717.

[65] Pride, S.J., (1995) Low Dimensional Homotopy Theory for Moniods, *International Journal of Algebra and Computation* **5**, 6, 631-649.

[66] Pride, S.J., Stöher, R. (1988) Relational Modules of Groups with Presentations in which Each Relator Involves Exactly Two Types of Generators, *Journal of the London Mathematical Society*, **2**, 38, 88-111.

[67] Rees S., Soicher, L., (2000) An Algorithmic Approach to Fundamental Groups and Covers of Combinatorial Cell Complexes, *Journal of Symbolic Computation*, **29**, 59-77.

[68] Rosentiehl P., Tarjan R.E., (1986) Rectilinear Planar Layouts and Bipolar Orientations of Planar Graphs, *Journal of Discrete and Computational Geometry*, **1**, 345-353.

[69] Schlag, M., Luccio, F., Maestrini, P., Lee, D.T., Wong, C.K., (1985) A Visibility Problem in VLSI Layout Compaction, In F.P. Preparata (ed.) *Advances in Computing Research, Volume 2*, Greenwhich: JAI Press Inc, 259-282.

[70] Serre, J.P., (1980) *Trees*, Berlin: Springer-Verlag.

[71] Shih, W., Hsu, W., (1999) A New Planarity Test, *Theoretical Computer Science*, **223**, 179-191.

[72] Soicher, L., (1993) GRAPE: A System for Computing with Graphs and Groups. In L. Finkelstein and W.M. Kantor (eds.) *Groups and Computation*, DIMACS Series in Discrete Mathematics and Computer Science Volume 11, American Mathematical Society, 287-291.

[73] Soicher, L., (1999) On the Structure and Classification of SOMAs: Generalizations of Mutually Orthogonal Latin Squares, *Electronic Journal of Combinatorics*, **6**, 1, #R32, (15 pages).

[74] Soicher, L., (2000) *Problems with Directed Graphs in GRAPE*, email to GAP Forum, dated 3rd April 2000.

[75] Stein, S.K., (1951) Convex Maps, *Proceedings of the American Mathematical Society*, **2**, 464-466.

[76] Tamassia, R., Tollis, I.G., (1986) A Unified Approach to Visibility Representations of Planar Graphs, *Journal of Discrete and Computational Geometry*, **1**, 321-341.

[77] Tutte, W.T., (1960) Convex Representations of Graphs, *Proceedings of the London Mathematical Society*, **3**, 13, 743-768.

[78] Voicu, L., Myler, H., Toma, C., (1998) Cloning Operator and its Applications, *SPIE Proceeding Series*, **3390**, 57-68.

[79] Wada, K., Nagata, Y., Chen, W., (1999) An Optimal Fault-Tolerant Routing for Triconnected Planar Graphs, *Lecture Notes in Computer Science*, **1665**, 191-201.

[80] Wagner, K., (1936) Bemerkungen zum Vierfarbenproblem, *Jber. Deutsch. Math-Verein*, **46**, 26-32.

[81] The WEST++ System, (`http://biomath.rug.ac.be/~westpp/`)

[82] Whitehead, J.H.C., (1949) Combinatorial Homotopy II, *The Bulletin of the American Mathematical Society*, **55**, 213-245.

[83] Woods, D., (1982) *Drawing Planar Graphs*, PhD Thesis, Stanford University.

Appendix A

Summary of Functions

▷ `3Presentation`($FpGroup, GenSet$) is a function that creates the 3-presentation of group defined by the presentation $FpGroup$, with a generating set of identity sequences given by $GenSet$. $GenSet$ is given by a list of lists such that each set of identity sequences are stored in a separate list. ▷ `AddPictureDisc`($sheet, x, y$) is called when a user presses the control key and the left mouse button on the sheet, $sheet$, at a point (x, y). It produces an XGAP dialogue box which requires the user to enter the number of arcs that will be incident on the disc and the discs label. If a relator information list has been constructed for the sheet only the label of the disc is necessary. The function then calls `PictureDisc` which produces the disc with centre at point (x, y).

▷ `AdmissiblePresentation`($data$) is a function contains methods for finding the fourth set of testwords of admissibility. A scheme for a presentation of an extension of $K = < \mathbf{x}; \mathbf{v} >$ by $Q = < \mathbf{a}; \mathbf{r} >$ is given by $data$. The record $data$ contains the following components: `3Presentation, GroupPresentation, FreeGroupExtension, PositiveEndo, NegativeEndo, Wr.` `FreeGroupExtension` is the free group returned by `AdmissiblePresentationWithGrp` when the first three sets of testwords are admissible. If scheme is admissible a presentation is returned; otherwise the testword the scheme failed upon is signalled.

▷ `AdmissiblePresentationsWithFpGrps`($data$) determines if a scheme for a presentation defines an extension of $K = < \mathbf{x}; \mathbf{v} >$ by $Q = < \mathbf{a}; \mathbf{r} >$. This function ascertains if all four

sets of testwords are trivial. *data* is a record with the following components: `3Presentation, GroupPresentation, PositiveEndo, NegativeEndo,Wr`. If scheme is admissible a presentation is returned; otherwise the testword the scheme failed upon is given.

▷ `AdmissiblePresentationWithGrp`(*data*) contains the method for computing the first three sets of testwords of admissibility. *data* is a record with the following components: `3Presentation, Group, FreeGroupExtension, PositiveEndo, NegativeEndo, Wr`. If all words equal the identity of `Group` then the function returns the free group of the presentation; otherwise the testword the scheme failed upon is signalled.

▷ `Arc`(*sheet*, *pdisc*1, *pdisc*2, *label*, [*arc*1, *arc*2]) draws the arc, α, between *pdisc*1 and *pdisc*2 on *sheet*. The arc numbers of α on *pdisc*1 and *pdisc*2 are given by *arc*1 and *arc*2 respectively, and *label* gives the generator that the connection point of *pdisc*1 depicts. The arrow of α is constructed with a positive orientation to *pdisc*1. The function finds the most appropriate positions for the endpoints of α on the circumference of *pdisc*1 and *pdisc*2.

▷ `ArcWithBends`(*sheet*, *pdisc*1, *pdisc*2, *label*, [*arc*1, *arc*2], [*values*]) produces an arc, *alpha* say, on *sheet* between *pdisc*1 and *pdisc*2, with label given by *label*. The connection points of α incident on *pdisc*1 and *pdisc*2 are given by *arc*1 and *arc*2 respectively. The coordinates of the endpoints of the lines that α consists of are given by [*values*], where the first set of coordinates points gives the value of the endpoint emanating from *pdisc*1, *values*[2] gives the coordinates for the remaining endpoint of the first line, and so on, until the last set of coordinates gives the endpoint of α on the circumference of *pdisc*2. Note that this function does not check that drawing α at this position does not contravene the rotation scheme of *pdisc*1 and *pdisc*2.

▷ `ArcWithCoords`(*sheet*, *pdisc*1, *pdisc*2, *label*, [*arc*], [*value*]) draws the arc, α, between *pdisc*1 and *pdisc*2 on *sheet* at coordinates points given by [*value*]. The arc numbers of α are given by [*arc*] and *label* gives the generator that the connection point of *pdisc*1 depicts. The arrow of α is constructed with a positive orientation to *pdisc*1. The function ensures that constructing α at these positions adheres to the existing rotation schemes for *pdisc*1 and *pdisc*2.

▷ `Basepoint`(*sheet*, *pdisc*) given that all arcs incident on *pdisc* have been constructed, `Basepoint` constructs the basepoint of *pdisc* on a basic corner.

▷ `ChangeBasepoint`(*sheet*, *pdisc*) changes the position of the basepoint incident on *pdisc* to another basic corner. If *pdisc* only has one basic corner, this is signalled and the basepoint does not change position.

▷ `CheckDiscs`(*FpGroup*, *RotationScheme*) ensures that the label of each disc in *RotationScheme* equals a relator of *FpGroup*.

▷ `DataStructure`(*FpGroup*, *arc*, *disc*) ▷ `DataStructure`(*FpGroup*, *arc*, *disc*, *file*) is a function which gives a produces a rotation scheme from a picture drawing. The arcs and discs of the picture are given in the lists *arcs* and *discs*. If *file* is given the data structure is saved to the file with filename *file*.

▷ `DefaultPictureMenu` options for the *Picture Menu* on a picture sheet.

▷ `DefaultControlsMenu` options for the *Shortcut keys* menu on a picture sheet.

▷ `Delete`(*obj*) deletes the picture object, *obj*, from the sheet it has been constructed on.

▷ `DiscDetails`(*FpGroup*, ρ, *Relatorinformation*) here ρ is a list of regions of a picture **P**. This function produces a record for every disc in **P** which stores the relator the disc represents, the orientation of the disc and the region where the basepoint of this disc lies.

▷ `DiscDetailsFromSheet`(*sheet*, *FpGroup*) here *sheet* contains a drawing of a picture **P**. This a function achieves the same as `DiscDetails` except the information of **P** is obtained from *sheet*. A record is computed for every disc in **P** which stores the relator the disc represents, the orientation of the disc and the region where the basepoint of this disc lies.

▷ `DisplayArcs`(*sheet*) produces a list of arcs that have been constructed on the picture sheet, *sheet*.

▷ `DisplayDataStructure`(*sheet*) produces the rotation scheme and relator information for the picture associated with *sheet*.

▷ `DisplayDiscs`(*sheet*) produces a list of picture discs that have been constructed on the

picture sheet, *sheet*.

▷ `DisplayExternalRegion`(*sheet*) produces a list of the external region of the picture associated with *sheet*.

▷ `DisplayLines`(*arc*) produces a list containing all lines of *arc*.

▷ `DisplayPresentation`(*sheet*) returns the group presentation for the picture associated with *sheet*.

▷ `DragArc`($line, x, y$) this is a function which deletes the relevant line, *line*, of an arc which has endpoints $\{(x_1, y_1), (x_2, y_2)\}$. In it's place it produces two new lines which have endpoints $\{(x_1, y_1), (x, y)\}$ and $\{(x, y), (x_2, y_2)\}$.

▷ `DrawPicture`(*FpGroup*, *RotationScheme*, *RelatorInformation*) draws the pictures over the presentation *FpGroup* which are represented by *RotationScheme* and *RelatorInformation*.

▷ `EnterGeneratingSet`(*FpGroup*, [*sheets*]) is a function that stores the generating pictures of FpGroup.

▷ `EnterPicture`(*sheet*, *FpGroup*) verifies that a given picture drawing is a picture over a presentation. If it is a picture and no basepoints have been drawn upon it, this function then constructs basepoints on the picture discs.

▷ `FpGroupOf3Presentation`(3*Presentation*) returns the 2-presentation of 3*Presentation*.

▷ `GeneratorsOf3Presentation`(3*Presentation*) returns the generators of 3*Presentation*.

▷ `GenSetOf3Presentation`(3*Presentation*) returns the generating set of 3*Presentation*.

▷ `HighlightArc`(*arc*) this is a function which changes the line width of the lines that construct *arc* and changes the outline colour to green.

▷ `HighlightPictureDisc`(*pdisc*) this is a function which changes the line width of picture disc (*pdisc*) and changes the outline colour to green.

▷ `IdentitySequence`(*FpGroup*, *RotationScheme*, *RelatorInformation*) this function produces the identity sequence of the picture represented by *RotationScheme* and *RelatorInformation*.

▷ `IdentitySequenceFromPicture`(*sheet*) this function produces the identity sequence of

the picture that has been drawn on *sheet*.

▷ `IntegerOfSquareRoot`(n) this procedure returns an integer, d say, where d is the the square root of n, rounded to the nearest integer.

▷ `IntegerOfSquareRoot`(n, m) this procedure returns an integer,d say, where d is the the square root of n/m, rounded to the nearest integer.

▷ `IsIdentitySequence`(S, G) is a function which determines if a given sequence, S is an identity sequence for the group G.

▷ `MakePictureMenu`(*sheet*) creates the *Picture Menu* on the sheet *sheet*.

▷ `MakeControlsMenu`(*sheet*) creates the *Shortcut keys* menu on the sheet *sheet*.

▷ `MoveArc`($sheet, x, y$) this is a function which moves the endpoint of an arc to a different position on the disc. It is called when a user selects an arc endpoint with 'shift left click', and the new position of the endpoint is given by the position of the cursor when the left mouse button is released. If this point is not on the circumference of the same disc, an error is signalled.

▷ `MovePictureDisc`($sheet, x, y$) this is a function which moves a picture disc to a different position on the sheet, *sheet*. It is called when a user performs a 'control right click' on a picture disc on *sheet*; the cursor is then dragged to a point (x, y) and the 'control right-click' released. The new position of the centre of the disc is then given by (x, y).

▷ `MoveArcLine`($arc, line, [[x1, y1], [x2, y2]]$) moves the line *line* of the arc *arc* so that the line now has endpoints given by {[x1,y1],[x2,y2]}.

▷ `OpenPictureEditor`($file, FpGroup$) opens the pictures over the presentation *FpGroup* stored in the file *file*.

▷ `PictureOfFpGroup`($sheet, FpGroup$) allows a user to enter a presentation, *FpGroup*, for a picture that has been/will be drawn on *sheet*.

▷ `PictureDisc`($sheet, x, y, r, label$) is a function which gives a produces a picture disc, with radius r, on *sheet* with a label given by *label*. The centre of the picture disc lies at (x, y).

▷ `PictureEditor`($name, width, height, FpGroup$) ▷ `PictureEditor`($name, width, height$) are functions which gives a produces an editing sheet with name $name$, with width $width$, and height $height$. If $FpGroup$ is given, the picture sheet will be used to create pictures over the group presentation, $FpGroup$.

▷ `PictureFromIdentitySequence`($3Presentation$) produces a representation (i.e the rotation scheme and relator information) of the pictures corresponding to the generating set of identity sequences of

▷ `PlaceBasepoint`($sheet, pdisc, [arc1, arc2]$) is a function which draws a basepoint between $arc1$ and $arc2$ of $pdisc$.

▷ `PlanarityTest`($RotationScheme$) investigates if the $RotationScheme$ of a picture represents a planar graph. If this is the case the regions of the picture are returned.

▷ `PlanarityTestWithRegions`($RotationScheme, \rho$) is a function for testing if multiple components of a graph with a given rotation system is planar. This function is the same as `PlanarityTest` except it takes previously computed regions, ρ, and the remaining regions of $RotationScheme$ as arguments. Again, if the graph is planar the regions of the graph are returned.

▷ `PropertiesOfDisc`($sheet, x, y$) is a function which gives a produces the disc (if any) that contains a given point (x, y) on sheet $sheet$.

▷ `ReducedFpGrpExtension`($FpGroup$) returns the reduced presentation of a group extension H. In other words the negative endomorphisms of $FpGroup$, the presentation for H, are removed.

▷ `RelatorLabels`($sheet, [[relator, label]]$) is a function which gives a stores the labels assigned to each relator. Each relator and it's label is given in a list of length two, $[relator, label]$.

▷ `RelatorsOf3Presentation`($3Presentation$) returns the relators of $3Presentation$.

▷ `SavePictureEditor`($sheets, file$) saves the picture sheets, given by $sheets$ into a file named $file$.

▷ `UserDragArc`($sheet, x, y$) this is a function which allows the user to drag part of an arc to

a different position on the sheet. The part of the arc is selected by 'shift right click' and the point of the 'bend' arc given when the user releases the mouse.

Appendix B

Inserting Discs into M

There are four ways in which discs are inserted into the Picture Visibility representation M. Let Δ be a disc in M. If no discs of Π_Δ are contained in M (i.e. Case (2) in Chapter 6), the discs in this local rotation are inserted into M using Type B.

As explained in Chapter 6, Case (3) occurs when one disc, Δ_h say, in the local rotation scheme for a disc Δ has been placed into M. If Δ lies at position $M[i][j]$ and Δ_h is located at $M[i_h][j_h]$ then the method chosen to insert the remaining discs of Π_Δ is given by the following criteria.

Type A: Arises if $i_h > i$.

Type B: This type is used for any other possibility.

Case (4) occurs when at least two discs of Π_Δ are contained in M. Let $\{\Delta_k, \xi, \Delta_l\} \subseteq \Pi_\Delta$ where ξ is an ordered set of discs that have not been placed in M. The discs Δ_k and Δ_l are contained in M and are located at $M[i_k][j_k]$ and $M[i_l][j_l]$ respectively. The position Δ is given by $M[i][j]$.

The first stage is to calculate the position of ϑ. If $j_l - j_k$ is even, set N equal to $\frac{j_l - j_k}{2}$; otherwise let $N = \frac{j_l - j_k + 1}{2}$.

1. If $i_k < i$ and $i_l < i$ and $j < j_k < j_l$ then $\vartheta = M[i-1][j_k + N]$. The discs of ξ are

inserted into M using Type A.

2. If $i_k > i$ and $i_l > i$ and $j < j_l < j_k$ then $\vartheta = M[i+1][j_l + N]$. The discs of ξ are inserted into M using Type B.

3. If $i < i_k$ and $i < i_l$ and $j_l < j_k < j$ then $\vartheta = M[i+1][j_l + N]$. The discs of ξ are inserted into M using Type B.

4. If $i_k < i$ and $i_l < i$ and $j_k < j_l < j$ then $\vartheta = M[i-1][j_k + N]$. The discs of ξ are inserted into M using Type A.

5. For all other possibilities $\vartheta = M[i][j]$. The method chosen to insert the discs of ξ into M is given by the following criteria:

   Type A: Arises if ($i \geq i_k$ and $i \geq i_l$ and $j_k < j_l$) or ($i \leq i_k$ and $i \leq i_l$ and $j_k < j_l$) or ($i \leq i_k$ and $i \geq i_l$ and $j_k < j_l$) or ($i \geq i_k$ and $i \leq i_l$ and $j_k < j_l$).

   Type B: Arises if ($i \leq i_k$ and $i \leq i_l$ and $j_l < j_k$) or ($i \geq i_k$ and $i \geq i_l$ and $j_l < j_k$) or ($i \leq i_k$ and $i \geq i_l$ and $j_l < j_k$) or ($i \geq i_k$ and $i \leq i_l$ and $j_l < j_k$).

   Type C: Arises if ($j \geq j_k$ and $j \geq j_l$ and $i_l < i_k$) or ($j \leq j_k$ and $j \leq j_l$ and $i_l < i_k$) or ($j \geq j_k$ and $j \leq j_l$ and $i_l < i_k$) or ($j \leq j_k$ and $j \geq j_l$ and $i_l < i_k$).

   Type D: Arises if ($j \leq j_k$ and $j \leq j_l$ and $i_k < i_l$) or ($j \geq j_k$ and $j \geq j_l$ and $i_k < i_l$) or ($j \geq j_k$ and $j \leq j_l$ and $i_k < i_l$) or ($j \leq j_k$ and $j \geq j_l$ and $i_k < i_l$).

Appendix C

Presentations for Groups of Order Less Than 32

The presentations for groups of order less than 32 are as given in the following table. If the group does not have a name associated with it, P_i is used, where i is the number of the presentation in the list.

Group Name	Relators
D_3	$R = x^2, S = y^3, T = (xy)^2$
D_4	$R = x^2, S = y^4, T = (xy)^2$
Q_2	$R = x^{-2}y^2, S = x^{-2}(xy)^2$
D_5	$R = x^2, S = y^5, T = (xy)^2$
D_6	$R = x^2, S = y^6, T = (xy)^2$
A_4	$R = x^3, S = y^2, T = (xy)^3$
Q_3	$R = x^{-2}y^3, S = x^{-2}(xy)^2$
D_7	$R = x^2, S = y^7, T = (xy)^2$
$C_2 \times D_4$	$R = x^2, S = y^2, T = z^2, U = (yz)^4, V = (xy)^2, W = (xz)^2$
$C_2 \times Q_2$	$R = x^{-2}y^2, S = x^{-2}(xy)^2, T = z^2, U = (xz)^2, V = (yz)^2$
D_8	$R = x^2, S = y^8, T = (xy)^2$
P_{12}	$R = x^2, S = xyxy^{-3}$
P_{13}	$R = x^2, S = xyxy^3$
P_{14}	$R = x^4, S = y^4, T = x^{-1}yxy, U = (x^{-1}y)^2$

Group Name	Relators
P_{15}	$R = x^4, S = y^4, T = (xy)^2$
P_{16}	$R = x^2, S = y^2, T = z^2, U = xyz(yzx)^{-1}, V = xyz(zxy)^{-1}$
Q_4	$R = x^{-2}y^4, S = x^{-2}(xy)^2$
$C_3 \times D_3$	$R = x^2, S = y^3, T = (xy)^2(yx)^{-2}$
D_9	$R = x^2, S = y^9, T = (xy)^2$
P_{20}	$R = x^2, S = y^2, T = z^2, U = (xy)^3, V = (xz)^3, W = (xyz)^2$
D_{10}	$R = x^2, S = y^{10}, T = (xy)^2$
P_{22}	$R = x^5, S = y^4, T = x^2y^{-1}x^{-1}y$
Q_5	$R = x^{-2}y^5, S = x^{-2}(xy)^2$
P_{24}	$R = x^3, S = y^2x^{-1}y^{-1}x$
D_{11}	$R = x^2, S = y^{11}, T = (xy)^2$
$C_2 \times A_4$	$R = x^3, S = y^2, T = (x^{-1}yxy)^2$
$C_2 \times D_6$	$R = x^2, S = y^2, T = z^2, U = (yz)^6, V = (xy)^2, W = (xz)^2$
$C_3 \times D_4$	$R = x^{12}, S = y^2, T = yxyx^5$
$C_3 \times Q_2$	$R = x^{12}, S = y^2x^{-6}, T = y^{-1}xyx^{-7}$
$C_4 \times D_3$	$R = x^{12}, S = y^2, T = yxyx^{-5}$
$C_2 \times Q_3$	$R = x^6, S = y^4, T = y^{-1}xyx$
D_{12}	$R = x^2, S = y^{12}, T = (xy)^2$
S_4	$R = x^4, S = y^2, T = (xy)^3$
P_{34}	$R = x^{-3}y^3, S = x^{-2}yxy$
P_{35}	$R = x^4, S = y^6, T = (xy)^2, U = (x^{-1}y)^2$
P_{36}	$R = x^{-2}y^2, S = x^{-2}(xy)^3$
Q_6	$R = x^{-2}y^6, S = x^{-2}(xy)^2$
D_{13}	$R = x^2, S = y^{13}, T = (xy)^2$
$B(2,3)$	$R = x^3, S = y^3, T = (xy)^3, U = (x^{-1}y)^3$
P_{40}	$R = x^3, S = x^{-1}yxy^2$
D_{14}	$R = x^2, S = y^{14}, T = (xy)^2$
Q_7	$R = x^{-2}y^7, S = x^{-2}(xy)^2$
$C_3 \times D_5$	$R = x^2, S = xyxy^{-4}$
$C_5 \times D_3$	$R = x^2, S = xyxy^4$
D_{15}	$R = x^2, S = y^{15}, T = (xy)^2$