Non-Visual Interaction with Simple Graphs

A thesis submitted to the University of Manchester for the degree of Master of Science in the Faculty of Science and Engineering.

2002

Andrew Brown

Department of Computer Science
Table of Contents

List of Figures and Tables .................................................................................................................. 5

Abstract .................................................................................................................................................. 7

Declaration ............................................................................................................................................. 8

Copyright ............................................................................................................................................... 9

Acknowledgements ............................................................................................................................. 10

1 Introduction ......................................................................................................................................... 11
   1.1 The Need for Assistive Software ................................................................................................. 11
   1.2 Non-Visual Interaction with Graphs ......................................................................................... 13
   1.3 Thesis Aims ............................................................................................................................... 15
   1.4 Contents of the Thesis .............................................................................................................. 15

2 Review ................................................................................................................................................ 17
   2.1 Problems Encountered when Understanding Complex Information Without Vision .......................................................... 17
   2.2 Converting Text to Sound: Screen Readers ............................................................................. 20
   2.3 Auditory Presentation of Mathematical Equations .................................................................... 20
   2.4 Presentation of Tabular Information ....................................................................................... 21
   2.5 Presentation of Diagrams ....................................................................................................... 22
   2.6 Summary .................................................................................................................................... 23

3 An Overview of Organic Chemistry ................................................................................................. 24
   3.1 The Valence Model of Bonding .............................................................................................. 24
   3.2 Functional Groups ................................................................................................................... 26
   3.3 Nomenclature .......................................................................................................................... 26
   3.4 Summary ................................................................................................................................... 27

4 Data Structure ................................................................................................................................... 28
   4.1 The Chemical Markup Language ............................................................................................ 28
4.1.1 Introduction to CML ................................................................. 30
4.1.2 DOM - Using CML Documents in Object-Oriented Applications .......... 32
4.2 Creating a Data Structure to Facilitate Browsing ........................................ 32
   4.2.1 The Structure of Information in Mathematical Equations ................ 33
   4.2.2 Transformation of Graph Data Structures ........................................ 34
4.3 Optimisation of the Data Structure .......................................................... 38
   4.3.1 Masking .................................................................................. 38
   4.3.2 Merging .................................................................................. 39
   4.3.3 Splitting .................................................................................. 41
   4.3.4 Automatic Optimisation ............................................................. 42
4.4 Summary ......................................................................................... 44

5 Algorithms ............................................................................................ 45
5.1 Identifying functional features ............................................................... 45
   5.1.1 Subgroup Isomorphism .............................................................. 46
   5.1.2 Ullmann's Algorithm ................................................................. 47
   5.1.3 Implementation of and Adaptations to Ullmann's Algorithm .......... 48
5.2 Identifying structural features ............................................................... 52
   5.2.1 Identifying Rings ..................................................................... 52
   5.2.2 The Balducci Algorithm ........................................................... 53
   5.2.3 Identifying Chains ................................................................. 55
5.3 Summary ......................................................................................... 56

6 The User Interface .................................................................................. 57
6.1 Navigation ......................................................................................... 57
   6.1.1 Connections .......................................................................... 58
   6.1.2 Zooming ................................................................................ 61
6.2 Interrogation ...................................................................................... 62
   6.2.1 General Information .............................................................. 62
   6.2.2 The Default Path Exploration ................................................ 63
6.3 The Audio-visual Interface ................................................................. 67
   6.3.1 The Diagram .......................................................................... 68
   6.3.2 The Menu Panel ................................................................. 71
6.4 Strategies for Using this Interface ........................................................ 72
List of Figures and Tables

Figures

1.1 A simple organic molecule represented in 2D: Ethanol................................. 13
3.1 2D representations of (i) methane and (ii) ethene.......................................... 25
3.2 Simple representations of ethanol................................................................. 25
3.3 Ethanoic acid.................................................................................................. 26
4.1 Summary of data handling process ................................................................. 28
4.2 Tree structure of a mathematical equation..................................................... 33
4.3 The structure of phenylalanine........................................................................ 34
4.4 Phenylalanine with numbered atoms.............................................................. 35
4.5 Data tree for phenylalanine............................................................................. 35
4.6 The functional groups of phenylalanine.......................................................... 36
4.7 Data tree for phenylalanine after identification of functional groups.............. 36
4.8 Complete data tree for phenylalanine............................................................ 37
4.9 Data tree for phenylalanine with structural features....................................... 38
4.10 Test molecule 3: (2E)-3-phenylpenta-2,4-dienoic acid................................. 39
4.11 Data tree for molecule 3................................................................................ 40
4.12 Data tree for molecule 3 after merging.......................................................... 40
4.13 Data tree for molecule 3 after splitting.......................................................... 42
4.14 Data tree for molecule 3 after splitting and masking..................................... 42
5.1 Use of extensible groups to simplify amino acid comparisons....................... 45
5.2 Search for carboxylic acids in 3-chloro-2-methyl-3-oxopropanoic acid............ 48
5.3 A fused ring system: naphthalene................................................................. 52
5.4 Collision classification in Balducci's network................................................ 54
5.5 Different possibilities for chain identification................................................ 55
6.1 The structure of cortisone............................................................................. 59
6.2 Exploring: group 1 - amino acid................................................................... 63
6.3 Exploring: group 2 - phenyl........................................................................... 64
6.4 Example of GUI............................................................................................. 68
6.5 Highlighting the CH group.................................................................69
6.6 Highlighting of carboxylic acid group.............................................70
6.7 Highlighting of primary amine group...............................................70
6.8 Menu while zooming in. .................................................................71
6.9 Menu giving group formula..............................................................72
7.1 Flowchart showing processes involved in preparing data for browsing........75
7.2 UML diagram for Group and related classes....................................76
7.3 Inheritance tree for Group objects..................................................78
7.4 UML diagram of connection classes................................................78
7.5 Information transfer in user interface..............................................80

Tables

3.1 Examples of chemical nomenclature .............................................27
4.1 Common elements in CML.............................................................31
6.1 Speech output when exploring cortisone..........................................67
Abstract

The need for providing information that is accessible to disabled users is being increasingly recognised; an act of parliament passed in the UK in 2001 places a statutory duty on schools and education authorities to improve in this respect. Graphs are a particularly common method for presenting complex information but are largely inaccessible to visually impaired users.

This thesis describes a system that enables visually impaired users to understand graphs. Graphs representing structures of organic chemistry molecules have been used as an exemplar class. These are represented on the computer by CML, chemical markup language, an application of XML. CML files are used to build a set of objects representing the molecule, and its atoms and bonds, using the document object model for CML.

The CML objects are used to create new objects designed to enable a hierarchical, tree-like, data structure to be generated. It is critical that the data are structured in such a manner that allows the user to be given control over the information flow, and in particular facilitates summarisation of parts of the graph. This is achieved by grouping atoms according to their membership of chemical functional groups or structural features in the molecule. Since some groups form parts of others this creates a hierarchy and the data structure becomes tree-like, with the entire molecule as the root and individual atoms the leaves. Intermediate nodes correspond to the groups. Methods for creating this data structure have been developed, as have techniques for automatically optimising the tree towards a deep narrow shape.

An audio-visual interface designed for browsing this data structure using synthesized speech and diagrams has been created. This allows users to move around the molecule and examine its parts at varying levels of detail, thereby understanding the graph.
Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.
Copyright

(1) Copyright in text of this thesis rests with the Author. Copies (by any process) either in full, or of extracts, may be made only in accordance with instructions given by the Author and lodged in the John Rylands University Library of Manchester. Details may be obtained from the Librarian. This page must form part of any such copies made. Further copies (by any process) of copies made in accordance with such instructions may not be made without the permission (in writing) of the Author.

(2) The ownership of any intellectual property rights which may be described in this thesis is vested in the University of Manchester, subject to any prior agreement to the contrary, and may not be made available for use by third parties without the written permission of the University, which will prescribe the terms and conditions of any such agreement.

Further information on the conditions under which disclosures and exploitation may take place is available from the Head of the Department of Computer Science.
Acknowledgements

I would like to thank both my supervisors, Doctors Robert Stevens and Steve Pettifer for their help and advice throughout this project. It was in no small part due to them that I enjoyed it so much.

I would also like to thank Jean, to whom I will very shortly be married, for her continual love and support. Her attributes as patient listener and critic to my often strange ideas were also most appreciated.
1 Introduction

This work is a limited investigation into the ways that computing technology can be used to facilitate the understanding of simple graphs by visually disabled users. A system has been developed to enable these users to browse, and thereby understand, graphs describing molecular structures. This chapter explains the needs for such a system and introduces the nature of the problem.

1.1 The Need for Assistive Software

Provision of quality education is, along with healthcare and law and order, generally regarded as a central role of government. Indeed the party currently in power in the United Kingdom, New Labour, has repeated the slogan "Education, education, education." innumerable times in an effort to convince the electorate that it is committed to focusing its efforts on this issue. Providing high quality education for students with disabilities, however, can be a particularly complex task.

The Royal National Institute for the Blind has estimated the number of visually impaired people in Great Britain for 1996 [1]. Their figures (based on research published in 1991 [2] and applied to population data from 1996.) indicate that there are 23,370 visually impaired people in the age group 0-15, and 161,710 in the age group 16-64. These figures are based on an overall population of 57,138,700. They show that there are a significant number of students and employees in this category.

In December 1999 the Disability Rights Task Force published a report entitled "From Exclusion to Inclusion" [3], in which a series of proposals were made for improving the lives of disabled people in the U.K. The task force examined many areas of life including housing, travel, education, access to goods and services, and employment. Chapter 4 dealt with education; one of it's recommendations was (emphasis added):

"Recommendation 4.10: Providers of school education should be placed under a statutory duty to plan to increase accessibility for disabled children to schools. This duty should cover both adjustments for physical access,
including those for children with sensory impairments, and for access to the curriculum."

This recommendation was one of many incorporated into a bill which passed into law as the Special Needs and Disability Act 2001[^4] on the 11th May 2001. The explanatory notes to this act[^5] say of the appropriate section (in Chapter 1 of Part 2):

"14. This Chapter places new duties on LEAs and schools (including independent schools and non-maintained special schools) in England and Wales and on Local Authorities (LAs), independent schools, self-governing schools and grant-aided schools in Scotland. The new duties are explained in the commentary on sections 11 - 16. What follows is an overview of the new provisions:

In England, Scotland and Wales
- a duty not to treat disabled pupils less favourably, without justification, for a reason which relates to their disability;
- a duty to make reasonable adjustments so that disabled pupils are not put at a substantial disadvantage compared to pupils who are not disabled (but there is no duty to remove or alter physical features or provide auxiliary aids and services); and

In England & Wales only
- a duty to plan strategically and make progress in increasing accessibility to schools' premises and to the curriculum, and in improving the ways in which written information provided to pupils who are not disabled is provided to disabled pupils."

It is the last part of this which has particular relevance for this work; the passing of this act has created a statutory duty upon schools to improve accessibility of information to visually disabled pupils.

This could of course be achieved by simply providing more teachers or classroom assistants to spend time with the students, but this is an expensive solution and a technological one is preferable. A piece or suite of software designed for non-visual
browsing of information would be a cheaper alternative, with the advantages of allowing more independence to the student. Furthermore, such software could have many applications within the workplace.

1.2 Non-Visual Interaction with Graphs

Graphs are widely used to convey all sorts of information in a quick and easily comprehensible manner - a quick glance often provides a great deal of information; "A picture is worth a thousand words" is the old saying. Typical uses include flowcharts, circuit diagrams, UML diagrams, entity-relationship diagrams, and so on: graphs are ubiquitous. Unfortunately their benefits are often either largely or completely lost for visually disabled users.

A common example of the use of graphs, in schools at least, is in chemistry, particularly organic chemistry. This important science is a basic necessity for giving students an understanding of the processes of life and is central to many industries in the United Kingdom including biochemistry, petrochemicals and pharmaceuticals. The subject will be introduced later, but essentially deals with the reactions of a huge number of different molecules based on carbon and hydrogen. These molecules take complex three-dimensional shapes but are generally represented as two dimensional graphs. A simple example is given below.

![Figure 1.1 A simple organic molecule represented in 2D: Ethanol.](image)

These graphs are generally simple and quite tightly constrained and are therefore an ideal class for which to develop a set of tools for browsing. These tools could then be generalised to deal with a wider range of graphs.
Technologies enabling visually disabled users to assimilate computer-based information are improving. Screen-readers are a common technology which will output as speech the text on a screen in a slightly interactive way. The Java Accessibility packages allow software to be developed which has user interfaces specifically enabled to be interrogated by such software. In general these users are fairly well catered for if the information they need is relatively simple, for example a text document. For more complex information, however, the task is much more difficult - how does one 'read' a graph from a document? There is a lack of available tools corresponding to this difficulty.

Specialist tools are available to provide Braille output representing some types of information, but these are specialist pieces of equipment and therefore both expensive for schools and restrictive for students. Audio output has the advantage that a sound card and speakers are commonplace, probably the norm for most computers. The only 'equipment' needed is the software, which is cheap and easy to distribute, once produced.

Work in the field of non-visual interfaces has been progressing. Research has been done into the presentation of complex information through an auditory interface. For instance Stevens \[6\] concentrated on mathematical information, particularly equations, such as the formula for solving a quadratic equation \(ax^2 + bx + c = 0\):

\[
x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}
\]

This research will be discussed in more detail in chapter 2. Interacting with graphs presents many more difficulties. Although complex, equations are often read aloud. It can be very difficult, however, even for experienced chemists, to describe a large molecule to one another without visual aids. Molecule graphs are perhaps more genuinely two-dimensional than equations, and it is this spatial complexity which is difficult to transform into sound. Speech is, after all, essentially one-dimensional.
1.3 Thesis Aims

The preceding two sections have described the need for systems which enable visually impaired students to conceptualise complex information and introduced some of the problems that arise when attempting to do so. The aim of this thesis is to describe a system that works on a simple subset of graphs, namely molecular graphs.

The problem can be summarised as two main areas:

1. How does one take the information present and transform it into the type of data structure that enables browsing? It has been shown that summarisation of the information is critical to avoid overloading the user with low level data. How may the information be grouped to enable summarisation, and how is the summary to be presented?

2. What are the best methods for control of the information flow: how can the user be in charge of the type of information received at any given time? The user must have the ability to examine different parts of the molecule and view the information at varying levels of detail.

It is intended that some of the principles used and developed to answer these problems may be applied to a more general solution for non-visual interaction with graphs.

1.4 Contents of the Thesis

Chapter 2 introduces the problems and some of the solutions proposed by others working in this field. Particular attention is paid to the work of Stevens, whose research into non-visual interfaces for mathematical information forms the starting point for this thesis.

This is followed in chapter 3 by an introduction to organic chemistry, which sets out to explain some of the basic concepts for the non-chemist. This chapter focuses on the representation of molecular structure and does not attempt to introduce reactions.
Chapter 4 describes how the structure of the information can be manipulated to facilitate non-visual browsing. The ideal structure is shown to be tree-like so that complex data can be summarised and the user can be given tight control over information flow. Methods for transforming the data structure for a molecule from an XML document to one enabling summarisation and control are described.

Achieving the desired data structure requires identification of functional groups and structural features in the graph representing the molecule. Chapter 5 discusses the algorithms used. Initially the problem of identification of functional groups is discussed; this can be demonstrated to be a form of subgraph isomorphism identification. Current algorithms for this problem are reviewed and Ullmann's algorithm described in detail. The second half of the chapter covers the identification of structural features; rings and chains. The smallest set of smallest rings (SSSR) is shown to be what is needed and algorithms for discovering the SSSR are introduced. The Balducci and Pearlmann algorithm is then explained in detail. Finally the adaptations made to this algorithm to enable identification of the smallest set of longest chains are described.

Chapter 6 deals with creation of a user interface for this data structure. Mechanisms for exploration of the tree representing the molecule are proposed, based on the idea that the user is currently located on a group of atoms and can navigate around connected groups, zoom in or out of the group to see a different level of detail, or interrogate the group to gain other information about it. A graphical interface is introduced and its coordination with the audio interface described.

Chapter 7 gives an overview as to how this system was implemented in Java. The overall structure is described, followed by brief introductions to some of the more important classes.

Chapter 8 is a discussion of this thesis; its contribution and limitations, and how the work may be progressed.
2 Review

Before describing the results of this research it is necessary to examine what has already been contributed to the field by other workers. There are several groups and individuals around the world with an interest in non-visual interfaces. After discussing the nature of the problems faced when trying to understand a graph without vision, this chapter will briefly describe some of their work with emphasis on the areas that overlap with presentation of graphs.

Initially the most common tool in use, the screen reader, will be introduced. These are widely used pieces of software capable of translating text-based information into sound. Following this will be a description of the findings of Stevens and his colleagues from their work developing an auditory interface for presentation of mathematical equations. This work is very much an extension of Stevens research. The rest of the chapter will be concerned with other types of complex information with brief discussions about presentation of tables, and the use of tones and haptic display to aid conceptualisation of diagrams.

2.1 Problems Encountered when Understanding Complex Information Without Vision

Some of the difficulties associated with trying to conceptualise complex information without visual aids were introduced in the previous chapter. Difficulties arise when the information no longer has the same dimensionality as the output medium.

Consider the example of a 2D matrix with large numbers in one corner which decrease as one moves toward the opposite corner. This type of feature would be picked up almost immediately by most visual readers. It would be much more difficult to detect, however if one were only allowed to see one number at a time, particularly if the matrix were just spoken from top left to bottom right. Important spatial relationships are hidden by compressing the information from a two-dimensional table to a one-dimensional stream of sound.
Vision has some tremendous advantages over the other senses for this task. Firstly it acts in three dimensions. Secondly it is very rapid; one may scan a page of data or a picture or view to 'get a feel for' the information. The eyes move quickly and precisely and the brain is capable of remembering general features, allowing a great deal to be inferred from a quick glance, particularly spatial relationships. Finally, as Stevens noted, the reader has control over his eyes and is capable of selecting what to look at. An area of information which is not fully understood may be re-read to complete the picture. This is not possible if the information is given as a continuous stream of speech.

Considering the limitations of sound as an output for multi-dimensional information, one may consider if the other senses may be used when vision is unavailable or limited. Smell may be immediately discarded as inappropriate, certainly for the foreseeable future! The sense of touch however offers interesting possibilities. There is the potential for three-dimensional representation of data, allowing users to literally 'get a feeling for' the data. Haptic representation may therefore be imagined to be an effective method for describing line or plane graphs. Research into this field will be briefly discussed in Section 2.5, although the sophisticated equipment required makes this medium unsuitable for general education at the moment.

The use of two or three-dimensional sound remains as an attractive long-term possibility and has been tested in a prototype screen reader [7]. The equipment required is less complex and expensive than that for haptic representation, although still currently unfeasible for schools. It must also be remembered that for a truly integrated education visually disabled students should have the minimum of specialist equipment.

For the time being then, the medium of choice is sound. As described above speech is essentially one-dimensional, but it is also possible to use non-speech sounds to convey information; tones have been used with some success to present line graphs (see section 2.4). To allow a user to understand a graph through the audio medium either speech or tones, or both, may be used. It is unlikely, however, that tones alone would be sufficient to convey the connectivity between nodes. Speech will therefore be essential for this application.
A graph can be defined simply as "a collection of nodes and edges". In particular the graphs representing molecular structure are connected graphs, i.e. there is a path (along edges) between any two nodes via zero or more other nodes. The atoms correspond to the nodes while the bonds are the edges.

The information that must be imparted to the user to enable their comprehension of the graph can be summarised as follows:

- The colour of each node.
- Which nodes are connected to each other, and by what colour edges.

The graphs may be cyclic and may have many connections from each node. This type of information is very difficult to transform into speech. It would be almost impossible to create a stream of speech describing a molecule atom by atom in a memorable and comprehensible fashion. One of the problems encountered is that there is often no clear start or finish point. Similarly having described one node there is no clear choice for which to select as the next; describing cycles without specifically explaining their existence and structure presents particular problems. The other major problem is that many molecules contain too many atoms to be memorable when simply listed.

Understanding these difficulties suggests two things:

- The information must be navigable; i.e. the user has control over what is described at any time.
- The facility for summarising parts of the graph must be provided.

These two features of control and overviews are critical, as has been demonstrated by many other research groups.

As was mentioned in the introduction the chemical graphs representing the molecular structure of organic molecules are an appropriate class for developing techniques for exploration of more general graphs. This is partly because they are tightly constrained connected graphs; the number of different node types are small, typically less than ten, while there are only three or four different types of edge. Connectivity is also limited with any node rarely connecting to more than four others. A further important and useful feature is the existence of formal nomenclature; part of a molecule may be given a name which describes accurately the configuration of nodes and edges within; this
simplifies summarisation. Chemical graphs by no means form a significant proportion of all graphs - the ubiquity of graphs and trees (a subset of graphs) was mentioned in the introduction to this thesis - but these factors make them a suitable test class and their investigation should allow discovery of the more important factors to be considered when developing non-visual interfaces for their understanding.

2.2 Converting Text to Sound: Screen Readers

Screen readers are commonplace and have been around for many years. They have developed from the simple ones used for the accessible command line interfaces such as DOS to more sophisticated programs designed for use with graphical interfaces as provided by Microsoft Windows and Macintosh operating systems.

Screen readers typically allow the user to navigate around the screen, normally with cursor keys or menu shortcuts, with features described as they are passed. Thus the user is able to use menus and buttons, and move around, for example, the document of a word processor. A simple screen reader for Macintosh computers is described by Alistair Edwards in chapter four of 'Extra-ordinary Human-Computer Interaction. Interfaces for Users with Disabilities.' [8]. Most modern systems work on the same principles.

2.3 Auditory Presentation of Mathematical Equations

In his research into the presentation of mathematical information to blind people, Stevens [6] made some important conclusions. Perhaps the most fundamental of these was that it is critical that the 'reader' has control over the information. They must be actively reading rather than passively being 'read at'. This arises largely from the problem that it is difficult to memorise more than a short amount of speech; deducing complex relationships between objects will therefore require some amount of flitting between them, almost certainly examining each object and relationship multiple times.

One aspect of control used by Stevens was the overview or glance. This was a high level look at the equation (or a part of it) to give a feel for its structure without the detail
of the individual expressions. Giving an overview was done by using musical sounds - 'Earcons' - to represent types within the expression, and necessitated discovery of structures. It was, however, considered important not to impart meaning while doing this; it is for the reader to impart mathematical knowledge to the equation. In creating the overview parts of the equation would be hidden - only represented by a summary - but could then be 'unfolded' later as the user investigates that part in more detail.

Stevens also spent much effort examining how best to use prosody - the variation in pitch, speed, and the use of pauses, etc. in speech - to maximise the information content and memorability of the spoken information. A set of rules for use of 'prosodic cues' was developed and found experimentally to be effective.

2.4 Presentation of Tabular Information

One piece of software designed for reading tables is TRIANGLE, which was written by Science Access Project at Oregon State University\[^9\]. The authors say of it:

"The primary purpose of TRIANGLE is to provide a workspace for reading, writing, and manipulating mathematics"

This is much more than a table reader as it also includes a maths/science word processor, a graphing calculator and a viewer for x versus y plots. The concepts behind reading mathematical equations and audio word-processing have already been introduced, but this system also deals with two dimensional information.

In Triangle tables are explored by one of two simple mechanisms. Either the information can be read one cell at a time, or one row at a time. Thus the user can navigate between cells or rows and read the information contained in the cell / row of his current position. This is effective at giving the information, although many relationships, like the one introduced at the start of section 2.1, would be difficult to detect. This is almost inevitable, however, without the reading software imparting meaning on the data.
Line graphs are similar to tables in structure (as long as the data used to create the graph are available) and complexity, although the fact that they are numeric simplifies the problem somewhat. Triangle, and other systems, use tones to present the information. In Triangle the user can initially find the labels, units and scales of the axes. It is then possible to move along the x-axis using a cursor; a tone is output whose pitch represents the y-value of the graph at that point. Additionally the cursor may be stopped at any point and quantitative information about the point given by speech. Thus an overview is gained by the tone pattern while detail may be gained by questioning actual values of, for example, maxima and minima.

2.5 Presentation of Diagrams

As was discussed above, once the information contains complex spatial relationships it becomes much more difficult to render into speech. Indeed only certain types of diagram are ever likely to be candidates for speech browsing: photographs or even line drawings are extremely difficult to describe even when making (unconscious) simplifications by interpreting the image. As a result most of the research in this area has concentrated on utilising the users sense of touch. This has generally involved specialist hardware producing a textured image. This might be via an array of Braille type dots in the simple cases, or for more sophisticated equipment the output might be produced using special 'swell paper’ or thermoform paper - these result in more continuous variations in texture. Once printed the user may feel the image to try to understand its structure. More recently hardware allowing dynamic display - i.e. an image is displayed only temporarily - has been developed [10].

There is also current research into haptic interaction, particularly by Prof. Brewster and his colleagues in Glasgow [11]. This technology uses hardware, and associated software, to allow the user to explore data via interfaces with force-feedback. For example, the user might navigate two dimensions using a joystick and resistance to the movement provides extra information (e.g. in addition to tones) about the nature of the data - a kind of third dimension.
2.6 Summary

Presenting graphs through non-visual media is a difficult task, particularly if spatial relationships need to be conveyed. There are many systems commercially available, or in development, for presentation of complex information. These are generally either confined to screen readers, which are only capable of effectively communicating text to the user, or more sophisticated systems using expensive technologies. The Mathtalk system devised by Stevens and colleagues with its speech and sound interface for reading mathematical equations is the closest to that required for this application and provides a suitable starting point for this work.
3 An Overview of Organic Chemistry

Organic chemistry is defined as:

"The chemistry of organic compounds; the chemistry of carbon compounds excluding the metal carbonates and the oxides and sulphides of carbon. Originally, it was the chemistry of substances produced by living organisms, as distinct from the inorganic chemistry of substances of mineral origin." [12]

And organic compounds are defined as:

"Chemical compounds containing carbon combined with hydrogen, and often also with oxygen, nitrogen, and other elements." [12]

Despite this apparently limiting definition the great majority of different substances found on earth are organic.

Organic chemistry is a very large area of study, but the understanding of a few simple concepts is sufficient to allow one to understand how graphs are used to represent molecules and how these graphs are constrained. This chapter attempts to summarise these concepts for the non-chemist.

3.1 The Valence Model of Bonding

Organic chemistry uses the 'valence model' of bonding as an abstract description of the extremely complex nature of chemical bonds. The majority of bonds in organic molecules are 'covalent', where two atoms each contribute an electron to the bond between them.

Multiple bonds are also possible; e.g. if a carbon and an oxygen each contribute two electrons to a bond it is said to have order 2, i.e. a double bond. Triple bonds are found, but bonds of higher order are not. Under this model carbon is most stable with a total bond order (a 'valence') of four, hydrogen one, and oxygen two. Thus a carbon atom
may have four single bonds to different atoms; two single bonds and a double bond; two double bonds; or any other combination as long as the sum of orders is four.

Using the valence model as a basis, organic molecules may be drawn in two dimensions as graphs. The nodes represent the atoms, while the edges represent the covalent bonds. Clearly both are coloured - the nodes to represent the different atom types, the edges to represent the different bond orders. The simplest organic molecule is methane, \( \text{CH}_4 \), shown below with ethene. Ethene is the simplest molecule with a bond of order > 1 and has a double bond connecting the two carbon atoms.

![Figure 3.1 2D representations of (i) methane and (ii) ethene.](image)

The carbon atoms may be connected into chains or rings, which may or may not contain multiple bonds. In molecules more complex than those shown above the notation is simplified so that hydrogens connected to carbons are implied (to make up the correct valence of 4 for the atom). Also carbons are not written explicitly; instead nodes representing \( \text{C} \) atoms are unlabeled. For example the molecule ethanol (Figure 1.1) could be drawn as any of the possibilities in Figure 3.2.

![Figure 3.2 Simple representations of ethanol.](image)

Similarly cyclohexane, a ring of 6 carbons each with two hydrogens attached, is drawn as a simple hexagon.
3.2 Functional Groups

Functional groups are those parts of a molecule which give it its reactivity. They are therefore usually the parts of most interest to the chemist. Generally these are collections of atoms which include 'heteroatoms' - atoms which are neither C or H. The '-O-H' atoms in ethanol are an example of such a group; this collection is known as an hydroxyl group. The hydroxyl group is itself part of some other groups such as the carboxylic acid group found in ethanoic acid (Figure 3.3).

\[
\begin{align*}
\text{H} & \quad \text{C} & \quad \text{O} \\
\text{H} & \quad \text{C} & \quad \text{O} \\
\text{H} & \quad \text{C} & \quad \text{O} \\
\end{align*}
\]

Figure 3.3 Ethanoic acid

The most common heteroatoms are oxygen (O), nitrogen (N), sulphur (S), phosphorous (P), and the halogens (F, Cl, Br, I). A list of common functional groups and their structures is given in Appendix 1.

The symbol 'R' on a structural diagram or in a formula is a common shorthand for 'anything'. Sometimes it implies a hydrocarbon group (chain, ring or combination), while in other circumstances it may also represent any functional group. In this thesis it has been used in its most general form; i.e. R represents any collection of atoms.

3.3 Nomenclature

A systematic method for naming organic molecules has been developed and formalised by the International Union of Pure and Applied Chemistry (IUPAC) \(^ {13}\). This provides rules for naming molecules so that the structure is known exactly. The rules generate names that become extremely long and complicated for larger molecules. Generating the structure from the name, and vice versa, is difficult, so much so that specialist software has been developed for this purpose \(^ {14}\).
An additional complication is the use of common names for many compounds. A large number of molecules have an old name which is still in common use, even in publications. This is particularly the case for some simple molecules such as ethanoic acid, which is commonly known as acetic acid, and complex biological molecules, such as cortisone. Table 3.1 below gives a few examples.

<table>
<thead>
<tr>
<th>Common name</th>
<th>IUPAC name</th>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethane</td>
<td>Ethane</td>
<td><img src="structure1.png" alt="Ethane Structure" /></td>
</tr>
<tr>
<td>Acetic acid</td>
<td>Ethanoic Acid</td>
<td><img src="structure2.png" alt="Acetic acid Structure" /></td>
</tr>
<tr>
<td>Phenylalanine</td>
<td>2-amino-3-phenylpropanoic acid</td>
<td><img src="structure3.png" alt="Phenylalanine Structure" /></td>
</tr>
<tr>
<td>Cortisone</td>
<td>11ß,17,21-trihydroxypregn-4-ene-3,20-dione</td>
<td><img src="structure4.png" alt="Cortisone Structure" /></td>
</tr>
</tbody>
</table>

Table 3.1  Examples of chemical nomenclature

3.4 Summary

Organic chemistry is the chemistry of molecules based on carbon and hydrogen. These molecules may be large and have complex three-dimensional shapes, which are represented in two dimensions by graphs. There are systematic naming conventions for molecules.
4 Data Structure

The importance of the structure of the data representing the information to be explored was mentioned in chapter 2. This chapter discusses this in more detail and describes the methods used to build a data structure and optimise it for browsing. The process involved in organising the data is summarised in the flowchart below.

Initially a drawing package is used to create a file representing the molecule of interest. For this Chemical Markup Language, an application of XML for representation of molecules on computers, is used. The file is then read by the program and suitable objects created to represent the molecule and its constituents. The next stage is to identify features in the data - in this case functional groups and structural features - enabling reorganisation of the data structure. Finally some optimisation techniques are applied to ensure the information about the molecule is structured in a manner best suited for non-visual interaction before allowing the user to browse the data. Each stage is discussed in sequence.

4.1 The Chemical Markup Language

In order for a graph to be explored using browsing software it is necessary to create or identify an existing format for storing it on the computer. Since many such formats already exist for molecules it is unnecessary to create a new one; the problem is reduced to selecting one of those currently in use. A suitable format ideally will be widely used.
and well known. It will also be easy to convert into from other formats. Since it is envisaged that schools will be using this system it is important that creating the files is both quick and easy, and can be done using cheap or, better still, free software.

Some of the many formats already in use include MDLMolfile, Sybil MOL2, JME, XYZ, SMILES, PDB and CIF. More recently the development of Chemical Markup Language (CML), an application of the extensible markup language (XML), has indicated a new era in the storage and transfer of chemical information on computers.

XML is a markup language designed to describe data. Data is stored in text format using Unicode characters and syntactic constructs known as tags are used to mark the starts and ends of elements within a document. For example a pair of tags may indicate that the text between them describes the title of the document. There is no set of standard tags; they must be created for the specific application and defined in a schema or a document type declaration (DTD). This ability to define a tagset is what makes XML extensible. Someone developing an XML markup language has the ability to decide what elements within the document are of interest and create tags for them.

CML is an application of XML that has been developed by Peter Murray-Rust and colleagues for containing chemical information [15]. A set of tags has been designed and a DTD created which specifies precisely how they must be used. The authors say of CML:

"Chemical Markup Language (CML) is an application of XML, the extensible markup language, developed for containing chemical information components within documents. Its design supports interoperability with the XML family of tools and protocols. It provides a base functionality for atomic, molecular, and crystallographic information and allows extensibility for other chemical applications. Legacy files can be imported into CML without information loss and can carry any desired chemical ontology." [15]

Although this is relatively new technology for chemists (this paper introduced version 1.0 of CML in 1999, although posters had been presenting describing its development as early as 1994) it is likely to become the standard.
From the point of view of this work CML is clearly the most suitable format to use. Being text based, storage, transfer and creation of CML files is easy. No special software beyond a simple text editor is required to create CML files describing molecules, although simple software is freely available to make this task much simpler \[16\]. CML files are also easily understood.

### 4.1.1 Introduction to CML

The following example of CML text describes the ethanol molecule from Figure 1.1, and was created by drawing the molecule in a special editor and exporting the molecule as CML.

```xml
<?xml version="1.0"?>
<!DOCTYPE molecule SYSTEM "cml.dtd" []>
<molecule>
  <atomArray>
    <atom id="a1">
      <string builtin="elementType">O</string>
    </atom>
    <atom id="a2">
      <string builtin="elementType">C</string>
    </atom>
    <atom id="a3">
      <string builtin="elementType">C</string>
    </atom>
  </atomArray>
  <bondArray>
    <bond id="b1">
      <string builtin="atomRef">a3</string>
      <string builtin="atomRef">a2</string>
      <string builtin="order">1</string>
    </bond>
  </bondArray>
</molecule>
```
CML describing ethanol.

The CML above starts with a line indicating that it is XML, then one describing where the DTD may be found. Then follows a series of nested tags describing the molecule. The molecule contains an atomArray and a bondArray which contain atoms and bonds respectively. Each atom has an id and an elementType, while each bond has the id of each atom it connects and the order of the bond. This describes the two C atoms and the O atom of the molecule.

The basic elements of CML which are most commonly encountered in this application are detailed in the table below (adapted from\cite{15}).

<table>
<thead>
<tr>
<th>element name</th>
<th>description</th>
<th>type of content</th>
</tr>
</thead>
<tbody>
<tr>
<td>molecule</td>
<td>generic container for any assemblage of atoms, bonds, or other molecules</td>
<td>element content, most normally an atomArray and a bondArray.</td>
</tr>
<tr>
<td>atomArray</td>
<td>a set of atoms</td>
<td>element content</td>
</tr>
<tr>
<td>atom</td>
<td>describes an atom</td>
<td>element content</td>
</tr>
<tr>
<td>bondArray</td>
<td>a set of bonds</td>
<td>element content</td>
</tr>
<tr>
<td>bond</td>
<td>describes a bond</td>
<td>element content</td>
</tr>
</tbody>
</table>

Table 4.1 Common elements in CML

It may be noted that the hydrogen atoms are not included in the CML example. This is somewhat dependent upon the software used to create the CML. A tag is available to
give the hydrogenCount of an atom, or (less common) the hydrogens may be included explicitly as atoms. Many other tags are available, for example to define the coordinates of the atoms in the drawing package, the formal charge of the atom, bond lengths etc. The DTD for CML is included in Appendix 2, and Appendix 3 contains 2D representations and CML descriptions of the test molecules used during development.

4.1.2 DOM - Using CML Documents in Object-Oriented Applications

A further advantage in using CML is the availability of tools for converting XML (and CML) files into computer objects using the document object model (DOM). This creates a tree of objects based on the hierarchy of tags in the CML document which may then be manipulated as desired by applications written in an object-oriented language. There is, in fact, a DOM created for CML[17] which is available on the internet[18]. This provides a set of Java classes allowing application developers to read CML files into objects representing molecules, atoms and bonds.

The objects created by CMLDOM are a CMLMolecule which contains a collection of CMLAtoms and CMLBonds. Each Bond references the atoms it connects and each Atom the atoms to which it is connected. It will be seen that these objects are not suitable for building the necessary data structure so new objects must be created that reference the originals. These are Group, Atom and Bond. The Group object represents a collection of atoms and bonds, e.g. a molecule.

4.2 Creating a Data Structure to Facilitate Browsing

It has been noted already that the structure of an equation was used by Stevens [6] to create a musical overview for the reader to 'glance' at. It is not so important that the overview was provided musically, but its existence was found to be very useful, and is likely to be even more so for more complex information such as graphs.
4.2.1 The Structure of Information in Mathematical Equations

The information in an equation can be seen to be structured as a simple tree. Figure 4.2 demonstrates this for the simple equation \( y = ax^2/(b + 2x) \).

![Figure 4.2 Tree structure of a mathematical equation.](image)

Stevens actually extended his work on equations to design a program named 'Treetalk' which used the design principles discovered in the maths work to allow non-visual exploration of trees.

Clearly this type of data structure allows hiding and summarising groups of objects, which in turn facilitates overviewing. An overview of part of the expression can be given at any level. Equally it should be obvious that a deep narrow tree is easier to summarise than a wide flat one; there are less terms for the reader to remember. The greater complexity afforded by the connections in more general graphs such as molecules leads to a rather different data structure. It can be shown, however, that it is possible to transform the graph to much more closely resemble the simple tree, and thereby facilitate overviewing.
4.2.2 Transformation of Graph Data Structures

On the face of it most molecules do not easily fit the tree structure that has been found to lead to useful overviews, and hence to a usable set of rules for browsing. The molecule graph itself is just a set of connected nodes with no start or finish points, and the object structure created by the CMLDOM is just a molecule containing a collection of atoms and a collection of bonds.

A mechanism for browsing molecules could be envisaged where the user is 'placed' on an arbitrary atom and could navigate around the molecule atom by atom. For example, a command could list all atoms connected to the current position. Another command could allow them to move to one of those atoms, and discover information about the bond connecting them. In such a manner the user could traverse the graph, walking around the atoms until the structure is understood.

Although this could be imagined easily for ethanol, but it would become more and more difficult to use as the number of atoms increases. For example, consider the amino acid molecule phenylalanine (IUPAC name: 2-amino-3-phenylpropanoic acid); the structure is given in Figure 4.3.

![Figure 4.3 The structure of phenylalanine.](image)

It would take a great deal of patience and concentration to understand the structure of phenylalanine by simply traversing the molecule atom by atom. The experience would be akin to trying to build a mental map of a maze whilst walking blindfold round it.

It can be seen, however, that the data structure for such a system is tree-like; a molecule node contains a collection of atom nodes. The summarising overview is the name of the molecule, or perhaps the number of each type of atom it contains, and the detailed
examination is the atom by atom navigation. If the atoms of phenylalanine are numbered (Figure 4.4), then the tree will look like Figure 4.5 (this does not indicate either bonds or hydrogen atoms).

![Figure 4.4 Phenylalanine with numbered atoms.](image)

![Figure 4.5 Data tree for phenylalanine.](image)

A quick look at this tree shows that it has a wide, flat structure, not the narrow, deep one that was concluded desirable for this application. It is necessary, therefore, to transform this tree so that it is more usable.

The concept of functional groups in organic chemistry was introduced in Chapter 3. These are collections of connected atoms which exhibit a certain reactivity. The hydroxyl group was seen in ethanol as an oxygen bonded to a hydrogen. Many more complex groups exist and a selection of the more common ones is given in Appendix 1. These collections offer a method for summarising parts of molecule - most chemists will understand the term 'hydroxyl group' and not need to investigate deeper to discover what comprises it.

If atoms within the molecule are grouped in functional groups the tree can be deepened and narrowed. A molecule will no longer consist just of atoms, but atoms and functional groups, which in turn may contain atoms and functional groups. If one
identifies the three functional groups present in the molecule phenylalanine, as shown in Figure 4.6, the tree can be transformed to a more suitable structure (Figure 4.7).

![Figure 4.6 The functional groups of phenylalanine.](image)

![Figure 4.7 Data tree for phenylalanine after identification of functional groups.](image)

It will be obvious that this new tree is an easier data structure to navigate. The molecule may be summarised as containing a phenyl group, an amine group, a carboxylic acid group, and two carbons. Navigation at this level is much simpler; instead of describing the molecule atom by atom we have:

"phenyl group connected to a carbon, which is connected to a carbon which is connected to an amine and a carboxylic acid."

This is evidently more comprehensible than atom by atom navigation. If a student is not yet aware of what a carboxylic acid group is, it is possible to investigate it atom by atom; since the tree is fairly narrow this is not an onerous task.
It may also be noted that the carboxylic acid group identified in phenylalanine above itself contains the hydroxyl group and a carbonyl group. One may also wish to define an amino acid group, in which case the tree could be improved further. This is shown in Figure 4.8.

![Complete data tree for phenylalanine.](image)

The molecule could now be described as:

"phenyl connected to CH₂ connected to amino acid"

Although this approach works well for a molecule such as phenylalanine, molecules which have large regions composed of carbon and hydrogen structures (rings or chains) will still be difficult to navigate. The atoms in these regions will not have any layers between themselves and the molecule node, and will therefore need to be investigated in the atom by atom manner. In order to avoid this undesirable situation it is necessary to identify structural features in the molecule, not just functional ones. Structural features include chains, rings and polycyclics (a collection of rings that are connected by sharing atoms).

The identification of structural features, and their inclusion in the object tree has some complicating side effects. It can be seen that phenylalanine contains a 6-membered ring (atoms 1-6) and a 3-membered chain (atoms 7, 8, 10). The atoms belonging to these features also belong to functional features. This changes the graph from a simple tree into a tree with cycles (Figure 4.9).
This would have a complex description if spoken, for example:

"phenyl group sharing 6 atoms with a six membered ring and connected to a 3 membered chain, which shares 2 atoms with an amino acid group"

Note that it would also be possible for the 6-membered ring to be above or below the phenyl group in this tree, rather than at the same level. It is the development of rules to deal with situations like this and keep the tree simple that will make a usable system for non-visual browsing of these graphs.

### 4.3 Optimisation of the Data Structure

The inclusion of structural features in the data tree has complicated it by introducing cycles and multiple super-groups. It is therefore necessary to optimise the tree in order to minimise this complication. This can be achieved by masking, splitting or merging certain groups

#### 4.3.1 Masking

Masking a group makes it invisible. For example if the '3-membered chain' in Figure 4.9 is masked then C<sub>7</sub> and C<sub>8</sub> will be seen to have only one super-group each ('phenylalanine' and 'amino acid' respectively). Similarly if the user asks what the molecule is composed of the groups listed will be 'phenyl', 'C<sub>7</sub>', and 'amino acid'. A further application of masking is to remove duplicate groups such as the phenyl group.
or the 6-membered ring in the phenylalanine tree above. If these two masking operations are applied the apparent data tree returns to that in Figure 4.8 with its simple description.

Since the functional groups are provided by a default set plus any that may have been defined by the user, it is important that these are not masked. It would not be user friendly to allow the user to define an amino acid group, then mask it so it is never seen. Masking can therefore only be applied to structural groups. Some structural groups, however, may be considered too important to be masked. This is the case with polycycles, as these are generally the dominant part of the structure. It is therefore also not possible to mask polycycles or their constituent rings.

4.3.2 Merging

The data tree may be narrowed by combining structural and functional groups. The result is another group which sits above the original two groups in the tree.

Consider test molecule 3 (Figure 4.10) and its data tree (Figure 4.11).

![Test molecule 3: (2E)-3-phenylpenta-2,4-dienoic acid.](image)

Figure 4.10 Test molecule 3: (2E)-3-phenylpenta-2,4-dienoic acid.
A speech description for this could be:

"phenyl connected to 5-membered chain which overlaps with a carboxylic acid group"

In this molecule the 5-membered chain may be combined with the carboxylic acid group. This results in the pentanoic acid group (N.B. For simplicity the nomenclature used here ignores the double bonds in the chain; the correct name would be penta-2,4-dienoic acid) which fits in the data tree as shown in Figure 4.12.

This could be described simply by:

"phenyl connected to penta-2,4-dienoic acid"

The description of the pentanoic acid group is complex however:
"5-membered chain sharing one atom with carboxylic acid group"

Although this merging does not remove the cycle in the tree it is making it more simple. The molecule may be summarised as being composed of a phenyl group connected to a pentanoic acid group. Previously this would have been a phenyl group connected to a 5-membered chain which shares an atom with a carboxylic acid group. Clearly this new summary is more easily understood. An experienced chemist will understand it directly while a less experienced user will have to interrogate the pentanoic acid group for more information. It will be noted that this strategy requires accurate use of standard chemical nomenclature.

Although the structure is improved, any investigation of the pentanoic acid group is still confused by the overlap between the carboxylic group and the 5-membered chain. This can be remedied by splitting and masking.

4.3.3 Splitting

The description of pentanoic acid as "a 5 membered chain sharing one atom with a carboxylic acid group" is both difficult to remember and confusing. More sensible would be something like "a 4-membered chain connected to a carboxylic acid group". This can be achieved by altering the tree.

If the 5-membered chain is split into a 4-membered chain and a carbon atom the resulting tree looks like Figure 4.13.
If the 5-membered chain group is then masked the tree appears as Figure 4.14.

The resulting description of the pentanoic acid group would be similar to:

"4-membered chain connected to carboxylic acid"

This new tree gives the clearest representation yet with no cycles and a narrow, deep structure.

4.3.4 Automatic Optimisation

Although the examples given above demonstrate effective methods for optimising the data structure, it is necessary to devise a strategy for applying these techniques to a tree
representing any molecule. The optimisation process must aid browsing of the molecule by simplifying the tree. Unfortunately the procedures of masking, merging and splitting are only effective under certain circumstances. It was therefore necessary to devise a set of rules defining when these methods may be applied. Furthermore, changes were made to the tree only if they resulted in a reduction in its complexity.

The algorithm used to determine which groups to apply the optimisations to is as follows:

Mask all structures with only one super-group and one sub-group.

For each chain:
- Try masking the chain.
- Try merging with any functional groups that share one atom with one end of the chain:
  - Split chain.
  - Create merged group.
  - Mask original chain.

Either leave, mask or merge depending on which gives lowest complexity tree.

For each (non-polycyclic) ring:
- Try masking the ring.
- Try merging with any functional groups that share one atom with it:
  - Either leave, mask or merge depending on which gives lowest complexity tree.

For each terminal functional group (i.e. is connected to only one other group):
- Try merging with first structural group it is connected to that is not terminal.
  - Either leave or connect depending on which gives lowest complexity tree.

This algorithm is relatively crude. For example only the first (non-terminal) structure found connected to a (terminal) functional group is tested for merging; others are ignored. It serves, however, to demonstrate the principle that automatic optimisation can be applied. It was also found to be effective.
4.4 Summary

A data structure suitable for allowing overviews and 'unfolding' of details has been shown to be important for enabling sensible non-visual browsing of information.

The graph representing the 2D structure of a molecule can be transformed by building hierarchical layers in the third dimension. This third dimension is a tree which may be made deep and narrow by identification of substructures in the molecule. Further optimisation may be achieved by combining and splitting groups. This can take place automatically by measuring the effect of each action on the complexity of the tree.

The resulting data structure is a tree where the root is the molecule and the leaves are the atoms. Intermediate nodes represent collections of atoms corresponding to chemical functional groups or structural features. Cycles are avoided where possible. This structure facilitates browsing by allowing the user to look at the molecule, or parts of it, at varying levels of detail.
5 Algorithms

In order to implement the data structure previously described, substructures in the molecule must be identified. The substructures can be classified into two groups; structural and functional. Structural features are those which describe the shape of the molecule; rings and chains. Functional substructures are those collections of atoms which impart chemical functionality to a molecule. This chapter describes the methods used to identify instances of both types of substructure within a molecule.

5.1 Identifying functional features

Functional groups are a basic chemical concept taught in schools from the introduction of organic chemistry. There are a great number of different functional groups defined \[13\], but many are rarely encountered, particularly in schools. Therefore it is desirable to define a simple but extensible set of functional groups to be identified in a molecule.

Identifying groups by hard-coding search algorithms in the program would not allow extensibility and would make reading the code difficult. The alternative of defining groups using CML and searching the molecule for matches is much more attractive. If a user wishes to browse molecules with a commonly occurring functional group which is not already defined, they may simply create a CML description of the group and it will be included in the search. For example a biochemist wishing to compare amino acids could define an amino acid group. This would separate the side chain from the parent structure giving the user a more understandable data structure for comparison.

Figure 5.1 Use of extensible groups to simplify amino acid comparisons. The amino acid group can be defined as (i), and will then be identified in the
different molecules (ii) and (iii) enabling comparison simply by browsing the side-chains.

The CML files describing the groups can be read into the computer using the same DOM as the main molecule. This simply leaves their identification as a case of subgraph isomorphism.

### 5.1.1 Subgroup Isomorphism

Subgraph isomorphism is the problem of discovering if one particular graph is a subgraph of another, that is all its nodes and edges can be mapped onto identical nodes and edges.

Identifying subgroup isomorphism in chemical graphs is a well researched field, indeed it is one of its most important practical applications. Companies doing research into creating new molecules, particularly in the pharmaceuticals industry, regularly search databases of millions of molecules to find those which contain certain features, and therefore may exhibit a desired reactivity. For these searches to be quick an efficient algorithm for subgroup isomorphism is essential.

The problem, and its major solutions were described in 1993 by John Barnard [19]. The difficulty is that the time required to solve the problem rises exponentially with the number of nodes (i.e. atoms), in fact it is an NP-complete problem. It is, of course, possible to solve the problem by 'brute force', that is checking every possible mapping for isomorphism. Many algorithms have been proposed which require much less computing time than the brute-force technique [20, 21, 22, 23, 24]. Barnard reported that work by Willet et al [25] had shown that Ullmann's algorithm [23] was the most efficient at that time (1988). Deficiencies in some of the other techniques were also described by Barnard. Further refinements of these algorithms have been made since, but the added complexity results in performance gains that are only significant in the huge searches performed in databases. It was therefore concluded that Ullmann's algorithm was the most appropriate to implement.
5.1.2 Ullmann's Algorithm

This algorithm is a back-tracking method with a relaxation-based refinement step. A
back tracking algorithm was first described by Ray and Kirsch[20] and is the basis for
many more recent substructure searching methods, including Ullmann's. The back-
tracking algorithm starts by mapping an arbitrary node from the query graph (A\textsubscript{N}) to an
arbitrary node on the molecule graph (B\textsubscript{M}). It then attempts to map each unmapped
neighbour of B\textsubscript{N} to each unmapped neighbour of B\textsubscript{M}. If successful the neighbours of
each neighbour are mapped, and so on until all nodes have been mapped. If the
mapping is unsuccessful at any point the algorithm 'back-tracks' to the last successfully
mapped node and tries a different mapping for it. If there are no different mappings
available the algorithm back-tracks again. If the original node A\textsubscript{N} has been tried on all
possible molecule nodes N\textsubscript{1...X} without success there is no isomorphism.

In Ullmann's algorithm this is implemented by building a binary mapping table M
between the query (A) nodes (rows) and molecule (B) nodes (columns) where a 1 in M\textsubscript{ij}
indicates that query node A\textsubscript{i} may map to molecule node B\textsubscript{j}. The first 1 in the first row is
taken as the arbitrary query node N\textsubscript{q}, and assumed to be the correct one. All other 1's in
that row are cleared. The first 1 in the next row is then selected in a similar manner, and
so on until we reach the last row. M is then tested against the connection tables of the
query and molecule nodes to check if the selected nodes are connected correctly. If so
an isomorphism has been discovered. If not the last row is reinstated and the second 1
selected. M is tested again, and the process continues. If none of the final row 1's give
a positive test, the previous row is reinstated. This back-tracking continues until an
isomorphism has been found or all the 1's in the first row have been checked, in which
case none exists.

So far this is simply a brute-force technique, but Ullmann introduces a refinement step.
This is performed each time a 1 (a node) is selected. The refinement tries to remove as
many 1's as possible from M. The 1's that may be cleared can be defined as follows. If
column k in row i of M is 1 we are supposing that B\textsubscript{k} may map to A\textsubscript{i}. Therefore any
neighbours of B\textsubscript{k} must also map to neighbours of A\textsubscript{i}. If this is not the case M\textsubscript{ik} may be
set to 0 without losing any isomorphisms. This can be tested by seeing if the logical
and of $M_i$ and $B_k$ equals 0. This is applied repeatedly until no more 1's are cleared. If at any point $M$ has a row with no 1's the refinement fails and the main algorithm backtracks.

The refinement step drastically reduces the number of unproductive mappings that need to be tested. It also removes the need for testing $M$ against the connection tables for isomorphisms since any $M$ where we have refined successfully after selecting a 1 on the last row must represent an isomorphism.

### 5.1.3 Implementation of and Adaptations to Ullmann's Algorithm

To implement Ullmann's algorithm so that it is capable of finding isomorphisms in graphs with coloured nodes (different element types) and coloured edges (different bond orders) it is necessary to devise a method for creating $M$. We define $M$ such that $M_{ij} = 1$ if atoms $A_i$ and $B_j$ have the same element type, and each bond has the same order and leads to an atom of the same type, otherwise $M_{ij} = 0$. For example consider the search for a carboxylic acid group in 3-chloro-2-methyl-oxopropanoic acid. These are shown with numbered atoms in Figure 5.2.

![Figure 5.2 Search for carboxylic acids in 3-chloro-2-methyl-3-oxopropanoic acid](image)

We can build the matrix $M$ for this search:

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$
Note that this does not include atom 4 of the search graph - the R group - this is defined as ‘not in the group’ using a new tag in the CML. This allows specification of an atom to be connected to (e.g. it must be a carbon) when the connected atom is not itself part of the functional group (e.g. consider differentiation of aldehydes and ketones). This matrix shows two possible matches for the carbonyl oxygen (search group atom 1 is the same as atoms 1 and 7 in the molecule), but only one each for the other atoms.

The algorithms described by Barnard, including Ullmann's are designed to test if there is at least one isomorphism of the query in the molecule. The interest for this work, however, lies in identifying all isomorphisms of the query. It was therefore necessary to modify slightly Ullmann's algorithm.

The modification is as follows. Once an isomorphism is found the atoms in the molecule that were mapped to it are recorded, then removed from the mapping table by clearing all 1’s in the corresponding column. The algorithm may then continue to find other isomorphisms. It should be noted that this means that only independent isomorphisms are found, i.e. those which share no atoms.

Two corrections were also made to the main algorithm as described by Ullmann. The corrected algorithm is as follows:

\[
\text{Step 1:} \quad M=M^0, \ d=1, \ H_1=0, \\
\quad \text{for all } i=1, \ldots, p, \set F_i=0; \\
\quad \text{refine } M, \text{ if exit FAIL then terminate algorithm;}
\]

\[
\text{Step 2:} \quad \text{If there is no value of } j \text{ such that } m_{0j}=1 \text{ and } f_j=0 \text{ go to step 7.}
\]

\[
M^d=M; \\
\quad \text{if } d=1 \text{ then } k=H_1 \text{ else } k=0;
\]

\[
\text{Step 3:} \quad k=k+1 \\
\quad \text{if } m_{ik}=0 \text{ or } f_k=1 \text{ then go to step 3;}
\]

\[
\quad \text{for all } j!=k \text{ set } m_{0j}=0 \\
\quad \text{refine } M, \text{ if exit FAIL then go to step 5;}
\]

\[
\text{Step 4:} \quad \text{if } d<p \text{ then go to step 6, else give output to indicate that an isomorphism has been found;}
\]

\[
\text{Step 5:} \quad M=M^d
\]
if there is no j>k such that \( m_{dj} = 1 \) and \( f=0 \) then go to step 7

go to step 3;

Step 6: \( H_d=k, F_k=1, d=d+1 \)
go to step 2;

Step 7: if \( d=1 \) terminate algorithm

d=d-1, M=M^d, k=H_d, F_k=0;
go to step 5;

The corrections were: 1. Changing Step 5 so that \( M=M^d \) comes before the test for other unused 1’s in the row. If this is performed the original way round then since the row has been cleared no other 1’s will ever be found and only the first possibility tested. 2. Changing the resetting of \( F_k \) so that it occurs after \( k=H_d \). This allows us to correctly keep track of which nodes in B have already been mapped.

The steps of this algorithm can be illustrated by following through the mapping matrix example above. M starts as:

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

Step 1 records this matrix and sets depth \( d = 1 \). The matrix F records which columns have been used: this is set to all 0’s. M is then refined; this requires use of adjacency matrices for the two graphs - these indicate which atoms are connected. They are the symmetrical matrices B and A below:

\[
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{bmatrix}
= \begin{bmatrix}
0 & 1 & 0 \\
1 & 0 & 1 \\
0 & 1 & 0
\end{bmatrix}
\]

The refinement loops through the neighbours of each carboxylic acid atom seeing if they are matched in the mapping matrix. For example atom \( A_1 \) (carbonyl oxygen in carboxylic acid) is adjacent to atom \( A_2 \) so any correct match of \( A_1 \) in the molecule (i.e. a
1 in row 1 of M) must also be adjacent to a match for A2. If we try the first possible match (column 1 - that is the wrong carbonyl) the AND of B1 with M2 is zero:

\[
\begin{array}{cccccccc}
B_1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
\text{AND M}_2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\end{array}
\]

This shows that although atom 1 in the molecule is a match for atom 1 in the carboxylic acid, it is not connected to an atom which matches atom 2 in the carboxylic acid. This is therefore an incorrect map and may be removed. This is done by clearing the 1, so M becomes:

\[
M =
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{array}
\]

The second possible match is now tested by performing the AND of B7 with M2:

\[
\begin{array}{cccccccc}
B_7 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\text{AND M}_2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\end{array}
\]

This is non-zero so the refinement continues by testing the other atoms. At the end of the refinement M has become:

\[
M =
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{array}
\]

Since refinement was successful the main algorithm continues with step 2. This and step 3 combine to traverse the first row until a 1 is reached, at which point all other 1's on the row are cleared and the matrix is refined. This will happen when the only 1 is reached in column 7, and as the matrix is unchanged since the last refine, refinement will be successful. Step 4 sends us to step 6 where we move down to consider the second carboxylic acid atom. The same process is repeated until we move to the third atom; once this is complete to step 4 the algorithm will identify the presence of an isomorphism with mappings:

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{array}
\]

Thus the procedure has correctly identified that atoms 1, 2 and 3 in the carboxylic acid are matched by atoms 7, 6 and 8 in the molecule. After these 1's are cleared from the
original M there two rows left that only contain zeros: restarting the algorithm to find
other isomorphisms will fail at the first refine - all isomorphisms have been found.

5.2 Identifying structural features

The structural features of interest are chains of carbon atoms, and rings.

5.2.1 Identifying Rings

Although a simple task if we have only independent rings in a molecule, ring
identification poses certain problems in systems where rings are fused, i.e. one or more
atoms are shared. In these cases, identification of all possible rings will lead to a set of
rings including some which are linear combinations of others. For example two fused
six-membered rings, as found in naphthalene (Figure 5.3), will have three rings; each of
the 6-membered rings and a 10-membered ring which follows the boundary of the two
rings. Clearly the simplest description of such a system is in terms of the two smaller
rings; also describing the larger will merely confuse matters. This set of rings is known
as the 'smallest set of smallest rings' (SSSR). It should be noted that a system may have
more than one SSSR.

![Figure 5.3 A fused ring system: naphthalene.](image)

A large number of algorithms for perception of an SSSR in a graph have been published
over the last 3 or 4 decades. Many of these have been developed for application on
chemical graphs. Downs et al. published a review of the more important algorithms in
1989 \[26\]. Research in this field has continued however, and new and improved
algorithms continue to be published \[27, 28, 29, 30\]. Fan et al. developed an algorithm that
works by selecting a 'root node', identifying the smallest ring it belongs to and
eliminating nodes which only belong to that ring \[27\]. With successive applications this
algorithm directly identifies the SSSR. This technique uses heuristics which Balducci and Pearlmann argue lead to an incomplete, inexact solution \[28\]. They use a very different technique based upon concurrent breadth-first searches. This algorithm will be discussed in more detail. Figueras \[30\] combined aspects of both these algorithms to produce a faster algorithm but since the benefits would be minimal for typical molecules implementing the extra complexity is not worthwhile for this system.

It was concluded that due to the exactness and efficiency of the algorithm, and the excellent description of its implementation in an object-oriented approach \[28\], Balducci's method would be implemented. The further refinements described by Mancini \[29\] to improve efficiency are unnecessary for this small application.

### 5.2.2 The Balducci Algorithm

To implement the concurrent breadth-first searches, Balducci maps the graph into a network of transceiver nodes (t-nodes) between which pass path messages. Each t-node matches a node in the graph and the path messages may pass between them if an edge connects the two nodes. Each t-node has two buffers for holding messages - a receive buffer which holds messages just received, and a send buffer which holds messages to be sent. Each path message contains the identities of the t-node it started from (n\text{first}), the first edge it travelled along (e\text{first}), the t-node it was last at (n\text{last}), and a binary edge-encoded path (beep) which notes the identities of all edges the message has passed along. The network is initialised with the receive buffer of each t-node containing a message from each of its neighbours.

The network then passes through a series of iterations where messages are passed on to all neighbours of a t-node except the one it came from. At each stage the messages come into the receive buffer and pass through a collision detector, where collisions between messages are identified and classified. A collision occurs when two messages arrive in the receive buffer of one t-node during the same iteration step. Four types of collision can be found.
Figure 5.4 Collision classification in Balducci's network. (i) No collision: The two messages do not have either common efirst or nfirst. (ii) Node collision: The two messages have a common nfirst but different efirsts. (iii) Inverse-edge collision: The two messages have a common efirst but different nfirsts. (iv) The two messages have common efirsts and nfirsts.

Clearly case (i) does not identify a ring and the messages are moved into the send buffer to be sent in the next iteration. Case (ii) identifies an even sized ring. In this case the two beeps of the messages are joined and the ring that the new beep represents is sent to the ring selector. The messages are then discarded. Case (iii) is similar, identifying odd-sized rings. Case (iv) does not identify any new rings since the smaller ring will have already been identified. One of the messages is discarded, the other is moved to the send buffer (an arbitrary choice).

The purpose of the ring selector is to examine the beep's representing the rings that it receives and determine if that ring is linearly independent from all rings so far identified. The rings received on the $c^{th}$ iteration will have size $s = 2c$ (even rings) or $s = 2c-1$ (odd rings), and all rings of a size $\leq 2c-2$ will have been identified already. The selector removes any duplicate rings then tests if any remaining can be reduced to a linear combination of the previously selected rings. If not then the ring is added to the list. The iterations stop when all $R$ rings ($R = E - N - 1$; where $N$ is number of rings, $E$ is number of edges, $N$ is number of nodes) have been identified.

This algorithm was adapted slightly so that the order of the nodes in a ring is known. The starting point is arbitrary but it is useful to identify the position of a node in a ring, for example when describing where features connect to a ring.
5.2.3 Identifying Chains

A new algorithm was developed to identify chains in the graph. This was designed to use the t-node network created by the ring selection algorithm. The initial step was to remove all nodes associated with rings, and all nodes not representing carbon atoms. A series of iterations is then performed in the same way as for ring selection, but instead of identifying collisions a node simply passes on any message received. If the node is terminal, however, this is not possible, and any messages received are passed to a chain selector. $E$ iterations are performed, where $E$ is the number of edges in the network.

The chain selector selects those chains which are independent of all others. It is designed to make selection with preference for longest chains. For example a 7-membered chain with a 2-membered chain attached in position 4 will be identified as such rather than as a 6-membered chain with a 3-membered chain in position 3.

This is achieved by initially ranking all the chains by length. Then starting at the top each chain is tested for overlap with any preceding. Only those found to be independent are kept. This results in a list of the smallest set of longest independent chains. Note that there are often different possibilities for this set; the one found is arbitrary.
5.3 Summary

Identification of functional groups has been reduced to a case of identifying subgraph isomorphism. Ullmann's algorithm has been identified as suitable for this application.

The algorithm for identifying rings (the SSSR) in the molecule is that first presented by Balducci and Pearlmann. This has also been adapted to allow identification of the smallest set of longest chains.
6 The User Interface

The user interface is the link between the data structure and the user. Having created and optimised a representation of the graph, a set of tools must be provided to enable the information to be browsed. The interface is the means by which the user obtains all information about the molecule and its design needs careful consideration. The interface developed for this system has not been perfectly designed, such a task would itself require several months work, but has instead evolved with the data structure. Nevertheless it incorporates some effective features which may be useful in the design of a more complete system. This chapter describes this interface.

The data structure has been developed, as described in Chapter 4, to facilitate control and summarisation. A hierarchical tree of groups has been formed which is as deep and narrow as possible. The user is considered to have a position in the tree representing the molecule and the intention is that they can navigate around the different groups to discover how these are connected. The summary of a group may not, indeed often should not, give sufficient information for a full understanding of its structure. It is therefore necessary to interrogate a group to gather more detailed information about it. Although closely related, the two aspects of navigation and interrogation will be discussed separately. Their coordination into a graphical and audio interface will then be described.

6.1 Navigation

The data structure previously described can be considered as composed of a series of layers. The level of any group is given by the number of layers between it and the base layer of atoms, e.g. if the atoms are defined as level 0 then the amino acid group in Figure 4.8 will be level 3. Navigation is the process of moving around the groups of a molecule at a certain level. Interrogation is finding more information about a group, which may involve moving down a level and navigating the sub-groups that compose the group.
The starting point for any exploration of a molecule will be the group representing the molecule itself. The user will then wish to find what groups make up the molecule. Two mechanisms enabling this have been designed. The default path, will be described in section 6.2.2, while zooming will be covered in 6.1.2.

### 6.1.1 Connections

Navigation is achieved by moving to connected groups. A simple command (typing the 'c' button on the keyboard) will give a numbered list of all the groups that are connected to the current one. Typing the number of one of these groups will move the user to that group and describe the nature of the connection.

Determining the connections to a group can be complex however. For example consider atom C7 in Figure 4.4: are its connections atoms C1 and C8, or groups amino acid and phenyl? A rule must be established to prevent the user from being pushed randomly up and down the tree. The connections are established first at the atomic level, i.e. the atoms directly connected to the group but not in it. The tree is then ascended up to the highest level group that is not the top-level and is independent of the current group. This group is the connection. Thus the carbonyl is connected to the hydroxyl and C8, not O12 and C8. The output given by the 'c' command while on the carbonyl would be:

1. bond to hydroxyl
2. bond to CH

The example above shows how information about the bonds may be obtained. Each connection includes a description of the type of bond connecting the two groups.

Simplification of this type of navigation is achieved by describing CH2 groups as part of the connection. These often form chains connecting functional groups and are very common. If one of the connection atoms is part of a CH2 then we simply move along the chain until something more interesting is found. For example the amino acid group in phenylalanine is connected only to C7, a CH2 group. Instead of describing the connections as:
1. bond to CH₂

we can say:

1. bond to phenyl via one CH₂ link

This reduces the number of groups at this level from three to two connected by a CH₂ link. Since such a CH₂ group can have only two single bonds there is no need to describe each bond order.

Further complication arises when rings are involved, particularly when they are members of a polycyclic. Consider the case of cortisone, whose structure is given below.

![Figure 6.1 The structure of cortisone.](image)

Describing how two fused rings are joined is complex, particularly so when there are three or more rings in the polycyclic and it becomes necessary to describe the relative positions. Correct IUPAC nomenclature for such systems is so complex that it cannot be understood without pen and paper; it is not suitable for this application. Instead the connections are explained by describing the numbers of the shared atoms. Since the indexing on two rings is different this requires 4 numbers to describe a single ring fusion. This will be difficult to visualise; these complex molecules are at the limit of what this system can deal with.

As an example when browsing the structure of cortisone zooming in to the molecule gives:

Zoom In
Then selecting the polycyclic and zooming in again:

Zoom In

1. cyclopentane
2. cyclohexane
3. cyclohexane
4. cyclohexa 5 ene

Selecting option 3 (the lower middle 6-membered ring in cortisone as drawn in Figure 6.1) the connections are spoken as:

Connections
1. bond from position 6 to methyl
2. positions 6 1 shared with cyclohexa 5 ene
3. positions 4 5 shared with cyclohexane

From this we determine that this ring is fused to two others with only a single edge separating the shared bonds (since atom 5 is shared with one ring and atom 6 the other). It is also seen that a methyl group is bonded to one of the atoms at the 'junction' between rings. These descriptions only give half the story, however, since we do not know which atoms of the other rings are shared. To discover this requires moving to them and asking the same question. For example, selecting connection 2 above:

Moving to cyclohexa 5 ene

Then asking for its connections gives:

Connections
1. position 1 shared with ketone
2. bond from position 4 to methyl
3. (Back) positions 4 5 shared with cyclohexane
The '(Back)' reminder tells us where we have come from. Now we know that positions 4 and 5 on the second ring are the same as 1 and 6 on the first.

To understand this structural system requires great concentration; although a general feeling for the molecule should be gained fairly rapidly, understanding the precise detail will take time and effort. Using the default path exploration described below simplifies this to a certain extent.

6.1.2 Zooming

Zooming is a simple process which allows the user to move up or down the object tree. The example of connections in cortisone given in the previous section illustrated zooming in from the molecule to a single ring. Zooming is achieved by using either the '+ ' or ' - ' keys to go to a level of greater or lesser detail respectively. When the key is pressed the user will be given a numbered list of options. Pressing the number of the desired option will take them to that group. For example if we return to the case of phenylalanine and the current group is amino acid (see Figure 4.8) the '+ ' command will give the following speech output:

1. Primary amine
2. CH
3. Carboxylic acid

While the ' - ' command gives:

1. Phenylalanine

Selecting a number that is not listed results in no action. If the user is at maximum zoom (i.e. currently on an atom) pressing ' - ' gives:

At maximum zoom.

Similarly if they are at minimum zoom (i.e. currently on the molecule).

When a group is selected to zoom to the focus changes and the move is described, for example:

Zooming to amino acid
6.2 Interrogation

Interrogation is the process of finding more information about the current group. There are two aspects to this: getting more information about the general characteristics of the group and finding what its sub-groups are.

6.2.1 General Information

There are three commands provided to give the user general information about their current group. The possible commands can be listed by typing 'p' for properties.

Firstly the name. This is obtained using the keyboard command 'n'. The name of the group is output. This is the name of the functional group (e.g. carboxylic acid), the element type if the group is an atom (e.g. C), or the name or description of the group if a structural group (e.g. 5-membered heteroatom ring, 4-ring polycyclic, or buta-1,3-diene). This is a reminder - the name of the group will be given when the user moves to it.

Secondly the identity. Each group has an identity number to differentiate different instances of the same type of group. This is given by the command 'i'.

Finally the molecular formula. This gives the numbers of each type of atom present in the group, and is given in standard format for these formulae. For example the formula for phenylalanine will be given as C_{9}H_{11}NO_{2}. This gives a great deal of information; from it one can infer the size of the molecule or group and get a rough idea how many and what types of functional groups it may contain. If the current group is not at the molecule level it will be connected to other groups. These are included in the formula as R groups, for example the formula for the amino acid group is RC_{2}H_{4}NO_{2}. One may thereby immediately know how many groups are connected.
6.2.2 The Default Path Exploration

This is considered to be the main method for investigating the components of a group. It is a single command which takes the user through the sub-groups in order. The key to this working effectively lies in describing this path in a comprehensible manner, a task which can become quite complex when it is cyclic but which is simplified by the creation of the deep, narrow data structure.

As discussed in Chapters 1 and 2, it is critical to give the user control over the flow of information. It would therefore be unhelpful to describe this path as a continuous stream of speech; instead it is broken up, the user able to move on when ready. This is best explained with an example. Consider again the case of phenylalanine. When the user first starts the program they are located on the molecule as a whole and will wish to explore the groups that it is built from. This is done by typing the command 'e', at which point there is the output

exploring

Then the first group is automatically selected; this group is highlighted in the diagram and it's name spoken:

![Diagram of phenylalanine](image)

Figure 6.2 Exploring: group 1 - amino acid.
amino acid

The system remains in this state until the user presses the space bar, which causes the next group to be selected:

Figure 6.3 Exploring: group 2 - phenyl.

*bond to phenyl via one CH2 link*

Another press of the space bar outputs:

End

And the diagram returns to highlight the entire molecule. Thus with one command the user can move through the molecule and discover that there is an amino acid group connected to a CH$_2$ group to a phenyl ring. They are then free to zoom into one of these groups to explore in more detail.

The following example shows how this works on a much more complex molecule - cortisone, which was introduced in the previous section.

Cortisone is based on a 4-ring polycyclic, with a two methyl groups attached, another bigger side-chain, an hydroxyl group, a couple of ketone groups, and one double bond. This is a large and complex molecule. When the CML file representing this molecule is
loaded and the molecule explored the sequence of speech and diagrams are as follows (not all diagrams are shown):

[open file] molecule: unknown

Exploring, ketone

1 atoms shared with cyclohexa 1 ene

bond from position 6 to methyl and
positions 1 6 shared with cyclohexane

positions 2 3 shared with cyclohexane

bond from position 1 to methyl and position 3 shared with ketone and positions 1 6 shared with cyclopentane

bond from position 3 to position 1 of 2-hydroxy-ethanone and bond from position 3 to hydroxyl
This can be written again without the diagrams to give a better impression of how a blind user would get this information:

```
e Exploring ketone
[space] 1 atoms shared with cyclohexa 5 ene
[space] bond from position 4 to methyl and
[space] positions 4 5 shared with cyclohexane
[space] positions 2 3 shared with cyclohexane
[space] bond from position 2 to methyl and
[space] position 6 shared with ketone and
[space] positions 2 3 shared with cyclopentane
[space] bond from position 3 to methyl and
[space] bond from position 3 to position 1 of 2-hydroxy-ethanone
[space] End
```

Table 6.1 Speech output when exploring cortisone.

This has not quite given us enough information to fully understand the structure, but that was not the intention. The molecule has been summarised: a 4-ring polycyclic with an unsaturated 6-ring then two cyclohexane rings then a cyclopentane ring, plus some side groups. Further exploration can now take place.

### 6.3 The Audio-visual Interface

The commands available have been described above with some examples of the speech output. The audio aspect of the interface has been combined with graphical output. The resulting window aids use of the system by users who still have some vision and allows non-visually impaired teachers or colleagues to work with it. Figure 6.4 gives an example of this window.
The window comprises three areas: in the bottom left is a panel which outputs all speech as text; above this is a picture of the molecule; on the right is the menu panel. The first is self-explanatory, the other areas are described below.

6.3.1 The Diagram

This area of the window gives a graphical representation of the molecule structure. The relative positions of the atoms are given in the CML file - most packages for producing CML are graphical and store atom coordinates automatically - these are scaled to fit the window. If no coordinates are included in the CML file no diagram is shown, instead this section of the window contains a message saying that coordinates were unavailable.

Each atom is represented by a filled circle. The colour represents the element type in accordance with standard chemical practice (such as used in plastic modelling kits), although only a limited set has been used here. These are as follows:

<table>
<thead>
<tr>
<th>Element</th>
<th>Colour</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>black</td>
</tr>
<tr>
<td>N</td>
<td>blue</td>
</tr>
<tr>
<td>O</td>
<td>red</td>
</tr>
</tbody>
</table>
Bonds are shown as a single line irrespective of bond order. Each atom is labelled with its group name, i.e. the element type with the number of hydrogens, and its identity (although the latter can be confusing for users and should probably be dropped), e.g. CH\(_2\) (#7).

An opaque mask is placed over the molecule. A window is cut into this to reveal all the atoms belonging to the current group. This shows the user which area of the molecule is being explored and could give great assistance to users with some vision. Additionally the window changes each time a group is spoken to indicate it's position. For example Figure 6.4 above shows phenylalanine with the amino acid group highlighted. If the next command is '+' to list the sub-groups the speech output will be:

1. CH (#8)

While this is being spoken the diagram will be:

![Figure 6.5 Highlighting the CH group.](image)

Then the synthesizer will speak:

2. Carboxylic acid

During which the diagram will show:
Finally as the last sub-group is listed:

3. Primary amine

The diagram will show:

Once all sub-groups have been summarised the diagram returns to highlighting the entire amino acid group. This system helps indicate the relative positions of each
group. It is applied whenever a group is spoken, being triggered by events fired by the Java Speech compatible synthesizer.

One final piece of functionality relating to the diagram is the facility to navigate the atoms using the mouse. If one clicks on the filled circle representing an atom the user’s current position moves to that atom. This is not particularly useful as it stands but this is an area to work on: the ability to zoom in on the diagram and navigate and interrogate using the mouse could prove highly useful for users with some vision.

### 6.3.2 The Menu Panel

This section displays all commands as a kind of tree. The tree expands and contracts as commands are given and changes slightly depending on the current group. The possibilities are listed and all keyboard commands may also be given by mouse clicks on the tree. Note that the behaviour of this panel has been different in Windows than Linux, although all commands are executed correctly.

The default state of the tree is as shown in Figure 6.4. When a command is given the appropriate node on the tree opens to give the options. For example requesting the subgroups by pressing '+' while currently on the amino acid group will speak the three groups and list them on the menu as in Figure 6.8.

![Figure 6.8 Menu while zooming in.](image)

If it is not possible to zoom in from the current group the menu changes to have the text 'At maximum zoom' instead of 'Zoom In'. If option 2 is selected and we zoom into the carboxylic acid, then asked for the formula, the menu would show:
And, of course, this formula is spoken.

6.4 Strategies for Using this Interface

Having provided a set of commands enabling the user to receive a variety of information it is useful also to suggest a strategy for best using these to quickly build a mental model of the molecule.

The default path exploration mechanism provides a technique for exploring the structure of a molecule at a given level, and it is anticipated that this will be the basis for the strategy. The first step in attempting to understand a new group should be to discover the formula. This gives a quick and easy picture of the size and composition. Following this the default path exploration will allow the user to understand the functional groups and structural features which form the group. Any further detail about how they are connected to each other can be gained by manual navigation around the groups. Zooming into and exploring a particular sub-group will allow the precise nature of its structure to be elucidated. Finally as one zooms back out, the groups may be summarised again by the default path to remind the user of the bigger picture.

6.5 Summary

A user interface has been designed to allow the molecule to be understood by navigating between its parts. Information is output via coordinated speech and graphics. A group of atoms is seen in overview as its IUPAC based name while mechanisms for moving between groups allow their relationships to be deduced. They can be also interrogated for more detail, for example by zooming in to the constituent
groups. The major tool for understanding the molecule at a given level of detail is the default path explore. This is a command that allows the user to be led through the subgroups of a group indicating their relationships.
7 Implementation

This chapter gives a brief overview of how the system described has been implemented. It starts with an explanation of the overall structure of the program. The classes designed to hold the information in the appropriate data structure are dealt with next. Following this is an overview of how the user interface operates.

Implementation has been done using Java 1.4.0_01 with development mostly in Red Hat Linux 7.

7.1 Program Structure

The flow of events was introduced in chapter 4. It can be summarised as: opening the CML file; creating atom and molecule objects using CMLDOM; creation of Group objects; identification of functional groups; identification of structural features; optimisation of the tree; display and navigation. This sequence is represented in more detail below.
Figure 7.1 Flowchart showing processes involved in preparing data for browsing

One of the features highlighted by this diagram is the flexibility allowed in selecting which functional groups are searched for. The files are created as CML files and another text file lists those files which the user wishes to identify in the molecule. There are two advantages to this system: extensibility and pedagogical flexibility. The former is obvious - an example of the advantage gained by the ability to define new groups was given in section 5.1 - but care is required when defining new groups, particularly with respect to use of 'R-groups'. The latter is perhaps less clear. One may, however, envisage a situation where a student is asked to identify carboxylic acid
groups in a molecule; in this case it would be necessary to remove this group from the search list - the exercise would otherwise be trivial and futile.

The length of the process before the GUI is displayed may cause problems with usability: a long wait with no visible action before the GUI is displayed or the speech engine starts is not desirable. Initialising the speech engine (this takes place when the GUI is started) is one of the slowest individual parts of this process, but is out of our control. This is an area for future improvement.

7.2 The Data Structure

The fundamental class is the one representing a group of Atoms (including a single atom) - Group. All nodes on the tree are a type of Group. The UML diagram below summarises this class.

Figure 7.2 UML diagram for Group and related classes
Figure 7.2 shows some of the main features of the Group class. Some of its more important attributes and methods are shown (not getter and setter methods). A Group object contains collections of other Group objects; its super-groups (those immediately above it in the tree) and sub-groups (those immediately below it); these references are sufficient to describe the structure of the tree. An Atom is also a type of group; in this case the GroupCollection subGroups will be empty. Other sub-classes of Group are described later.

The Group also contains two collections of bonds; internalBonds and externalBonds. The externalBonds collection contains all bond objects that connect the Group to other Groups. The internalBonds collection contains all bonds that connect the subGroups within the Group.

Only a selection of the methods for this class are shown. There are methods for getting the attributes such as name, id, etc., as well as methods for finding other information about the Group - is it terminal? (i.e. is it only connected to one other Group?), is it asymmetric? (i.e. do the external bonds connect from different atom types; compare esters with acid anhydrides, etc.) and so on. There are also methods for getting the sub- and super-groups, although these are complicated by masked Groups; sometimes one needs to find the true sub-groups while at other times (for example when navigating) only the unmasked Groups are wanted.

The UML diagram in Figure 7.2 shows Atom as a sub-class of Group. This is because Atoms exhibit all the characteristics of Groups (albeit with empty internalBond and subGroup collections) plus some other attributes. It will be noted that one of these is the CMLAtom object that is created by the CMLDOM; this gives access to all information specified in the CML document which is understood by CMLDOM. There are also sub-classes designed to represent other specific types of Group. The following class diagram demonstrates the inheritance hierarchy under Group.
Figure 7.3 Inheritance tree for Group objects

The classes Atom and Structure are obvious extensions of Group; there are extra attributes and methods such as formalCharge or atomOrder (this holds information about the order of the atoms in the structure). Although Polycycle is really a type of Structure it needs to be treated differently; giving it a separate place in the hierarchy is one method for achieving this.

Of the types of Structure, Rings, Chains and Polyrings (a ring which is a member of a polycycle) are also obviously placed in the hierarchy. A CutStructure is a Chain which has been cut to simplify the tree. A MergedStructure is a Group formed by the combination of a Structure with a functional group.

Three other classes represent the connections between Groups. The classes and their attributes, but not the methods, are shown in the following UML diagram.

Figure 7.4 UML diagram of connection classes.
SimpleConnections are used to describe connections where the Groups are joined by a bond or a series of -CH₂- links; these have additional attributes referencing the Bond and specifying the atoms at either end. Overlaps are used where the Groups share atoms, such as two rings in a polycycle, and have an Atom array and a Bond array referencing all shared Atoms and Bonds. There are methods for getting all relevant information and related objects for each class.

7.2.1 Sub-structure Searching Algorithms

Ullmann's algorithm for subgraph isomorphism is implemented as follows. A SubgroupSearcher is created for the molecule. This object coordinates the searches for functional groups and assigns Groups a position in the tree. Its search method is called for each type of group; this creates an UllmannSearch object which performs the algorithm. A single UllmannRefineAlgorithm object is created for the search - it is sent a new mapping matrix for each refine. The UllmannSearch search method returns a Vector of atom collections, each one containing the atoms of a matching group. A Group is created for each collection and it is positioned in the tree.

A StructureSearcher performs a similar role to that of the SubgroupSearcher when identifying structural features, except that one instance of this class is used to find all rings and chains. Simply creating a StructureSearcher and calling its search method will identify all rings and chains, create Group objects and insert these into the tree. This object uses a RingSearchEngine object to perform the searches. This creates the network of transceiver nodes and populates it with path messages, as described by Balducci (the RingSearchEngine corresponds to his ring selector). It is responsible for performing the independence tests for new rings, and will adapt the network for identifying chains. The StructureSearcher uses the SubGroupSearcher method for assigning the structures a position in the tree.

7.3 User Interface

The following figure gives a diagrammatic representation of the roles of the different objects in the user interface. A rectangle represents an object and a rounded rectangle a method. Inner classes are shown as rectangles within another class, if they are
parallelograms the actions are performed by an anonymous inner class. Ovals represent inputs from the user.

Figure 7.5 Information transfer in user interface.

To summarise this diagram normal inputs (i.e. those controlling the program, not exiting or controlling the window) can be seen to be handled by one of two objects. Clicking on an atom in the diagram is handled by an anonymous inner class which sends a message to the MoleculeFrame to change the focus group. Selecting a node on the menu tree can be done by mouse, which creates an event handled by an anonymous inner class of the OptionsTree. These events can also be created by handling key strokes in the ExplorerKeyHandler of the OptionsTree; these cause direct changes to the tree which fire TreeChangedEvents. Changes to the tree by either method will send an object or a string to the Frame for speech output, and may also invoke its setFocusGroup() method.

The MoleculeFrame has an anonymous inner class that listens for SpeechEvents. When one is received the Group that is being spoken is highlighted by invoking the appropriate method of the MoleculeFrame object.
7.4 Speech Synthesis

The speech engine used was FreeTTS, a speech synthesiser from Sun Microsystems which is written entirely in Java and is freely available [31]. This provides partial support for the Java Speech Application Programming Interface (JSAPI) but not, unfortunately, the Java Speech Markup Language (JSML). JSML is an XML based markup language that allows a string to be marked so that pronunciation, prosody, etc. may be defined. Marks may also be inserted to generate speech events. This would be particularly useful for this application. JSML text is accepted by the FreeTTS speech engine, but the tags are ignored.

Since the Group and Connection classes implement the Speakable interface these can be sent directly to the speech engine, as long as this supports JSAPI. A class implementing this interface has a method getJSMLText() which returns a string to be spoken. Therefore anytime a Group needs to be described it may be sent directly to the SpeechEngine. This has the further advantage of providing a work-around to generate speech events (although events cannot be triggered by JSML markers, the FreeTTS software produces speakableStarted and speakableFinished events) and know exactly when the object is being spoken: thus we implement highlighting.

7.5 Java Accessibility

The designers of Java have allowed for disabled users by developing Java Accessibility. This is a method for standardising information about GUI components so that other accessibility software may interact with them. Each GUI component has an accessible name and description; both are set manually in the code - the description is the ToolTipText of the component, the name is set by the method component.getAccessibleContext().setAccessibleName("name....").

This software is designed to be 'stand-alone' in this sense: it is accessible to blind users without other assistive software. The output of a screen reader while running this program could prove very confusing. Nevertheless the GUI components have been built to be accessible.
8 Discussion

This chapter summarises the work described in this thesis and what was achieved. It discusses some of the limitations and how they may be overcome. Areas for future work are suggested, both for improvement of the system described and for generalisation to other graphs.

8.1 Summary of this Research

This thesis has described the development of a system to enable blind and visually impaired users to conceptualise graphs representing the structure of chemical molecules.

Previous research into non-visual presentation of complex information has shown the need for an appropriate data structure. This can be demonstrated to be tree-like, with a deep and narrow form and a minimum of cycles. A mechanism for transforming the complex, often cyclic, graphs representing molecular structure has been developed.

Molecular structure can be considered as a two or three dimensional graph. This is difficult to understand by simple node to node navigation, so an extra dimension was added; this dimension has the desired tree-like structure. This was achieved by grouping atoms, then grouping groups and leads to a tree with the atoms forming the bottom level, and the molecule the top. The description of a group summarises the atoms within it, and their relationships with each other.

Atoms were grouped according to their membership of chemical functional groups - well defined groups of atoms, generally with interesting chemical behaviour - or structural features - chains or rings of atoms. A modified version of Ullmann's algorithm for sub-graph isomorphism was applied to identify functional groups. The groups searched for form an extensible (or reducible) list of CML files. Balducci and Pearlmann's algorithm for identification of the SSSR was applied to identify rings and modified to allow identification of chains.

Further transformation of the tree is performed by an automatic optimisation routine. This attempts to form combination groups by joining functional groups to structural
features, or simplify the graph by hiding complicating structures. These actions are
only performed if they result in a simplified tree.

Standard chemical nomenclature, as defined by IUPAC, has been used as much as
possible to name groups. This allows experienced chemists to understand the overview
of a group without needing to explore its structure in more detail. Novices will,
however, be able to investigate the structure of any group down to the atomic level.

A user interface has been designed that integrates the visual and audio media. A
diagram of the molecular structure is displayed where possible (i.e. where the CML file
gives atomic coordinates) and atoms or groups are highlighted when being spoken.
This gives extra information to users with sufficient vision, speeding up their
comprehension of the data.

The principle of user control was crucial in the development of the interface. A user
must first see a summary of the molecule or group before being able to 'unfold' its
structure and investigate the details. The concepts of navigation and interrogation
describe the idea behind the design. A user is considered situated on a group and may
interrogate it to discover its structure in more detail, or navigate to other connected
groups. They are, in essence, moving around the groups of the tree. An important
facility is the 'default path explore' where the user is led around the sub-structures of the
current group: this is the main mechanism for interrogation. Other summarising
features are the name and the formula.

8.2 Limitations of this Research

Apart from some minor bugs the main limitations of this work can be grouped into two
areas: those relating to generating a suitable data structure and those regarding the user
interface.

8.2.1 Data Structure

The data structure could be simplified further. As was described in section 4.3.4, the
further optimisation procedures are crudely applied. For example, only the first
structure connected to a functional group is tested for merging. Although this leads to a simple data structure for most molecules tested, performance could be improved by making this procedure more sophisticated. For example, one possibility is to use ranked groups: IUPAC nomenclature rules rank functional groups to determine which is used for the suffix in the name (the most important group) and which for the prefixes. Giving the groups defined in CML files a rank would allow ranking to be used in this system. There are, however, some difficulties associated with this; the main one being the lack of rank for large 'unofficial', but useful, groups such as amino acid.

It may also be possible to do more connecting and merging in large molecules. Since the data structure achieved was worthwhile and comprehensible even for the more complex of the test molecules, this may prove ineffective.

### 8.2.2 Nomenclature

The nomenclature for groups used in this work is based on the IUPAC recommendations. There are some complications with this approach, largely arising from the complexity of the rules. On the one hand the rules can be so complex that it is difficult to apply them in such a relatively simple piece of software - this work is not intended to be a name generator - while on the other the name might be simple to generate but still complex. In the second instance the complexity of the formal name diminishes its capability to summarise.

Although it is believed that even a complex name can summarise; the user may simply gather that the group is a chain or a ring, whether it is saturated or not, and what functional groups are in it; there is scope for better summarisation here, perhaps with the formal name of a group as a second port of call for information.

Another difficulty with the application of IUPAC rules for nomenclature is that they are not really designed for independently describing parts of molecules. In this work each part has been described as if it is a separate entity. For example a -CH₂-CH₂- chain which is attached to another group at each end will be described as "ethane" (ethane is the molecule CH₅-CH₅), while CH₃-CH₂- will be correctly described as "ethyl". Experiments are required to determine if this is correctly understood by users.
8.2.3  User Interface

Some of the features of the user interface have only been implemented in a demonstrative way, that is to show the potential without full functionality. For example the ability to interact with the diagram using the mouse is restricted to simply moving the focus to the selected atom. Similarly use of the diagram is intended for visually impaired users who are not blind, but there are no facilities for expanding the picture or selecting more suitable colours etc.

8.2.4  Molecule Complexity

The system described is not one which will allow a blind user to understand the structure of any molecule imaginable. Although theoretically capable of being used on arbitrarily complex molecules its efficacy is limited on large and complex ones; consider for example the molecule $C_{60}$, which is a network of 5 and 6 membered carbon rings with the same structure as a football. The example of a steroid given in section 6.2.2 probably represents a realistic limit, although perhaps cortisone is slightly too complex - the difficulties in conceptualising the spatial relationships between the fused rings should be clear.

8.3  Future Work

8.3.1  Improving spatial understanding.

The system described above allows the user to understand connectivity and build a mental image of the molecule. This can be very difficult, however, particularly if they are unable to use the diagram. In order to overcome this it may be useful to give clues about the position of a group.

There are several possibilities for doing this. Firstly, when moving from one group to another the direction travelled may be described, e.g. "moving NW to carboxylic acid". Secondly a user may be able to ask for the position of a group. Providing the CML file gave atomic coordinates it would be simple to provide a coordinate for the centre of gravity of any group. This is a simple and potentially useful facility, although some
difficulties may arise when describing straight chains; a centre of gravity is not sufficient for a good description of position.

Another mechanism would be to use two or three-dimensional sound: when describing a group the speech would be 'positioned' depending on where the group is. This has the major disadvantage of requiring expensive equipment and is therefore not currently suitable for school use.

8.3.2 The User Interface

There is great scope for improving the visual aspects of the user interface. The diagram of the structure could be made much more accessible to those users who have some vision. There should be scope for zooming in on the picture and seeing more information at a large scale. Further improvements would include accurate depiction of bond order (the diagram currently displays all bonds as a single line), and more sophisticated facilities for mouse interaction.

There would also be benefits from designing a command language from scratch. The basic principle of mnemonic key commands appears to be useful, although a significant increase in the number of command options would complicate this approach.

The output of information has great scope for development. The system described uses only speech and vision; non-speech sound has not been investigated. Since this was found by Stevens to be useful for summarising parts of equations in the form of 'Earcons' it ought to be studied. In conjunction with this, the degree of control over speech ought to be further investigated. Experimentation may show that information may need to be presented in smaller chunks, for example when listing the connected groups.

Further development of information output could involve integrating other non-speech media. The most obvious is probably enabling Braille output, either permanently to paper or to a temporary output device: there are rules defined for representing molecular structure in Braille. This is, however, an almost orthogonal approach to the use of speech and would use few of the techniques developed here. A Braille output program
could be entirely independent, but a package providing both would give the user much more flexibility. Similarly a picture of the molecule could be output as an embossed image.

Even the current use of speech has potential for improvement. The benefits of appropriate use of prosody to infer structure on information and aid its memorability have been demonstrated for other applications but not investigated here. A fully JSAPI compliant speech engine would have access to all JSML elements - this would allow easy fine grain control of speech output.

8.3.3 Extension to more general graphs

The intention of this work was to explore methods for improving accessibility of complex information to visually disabled users, particularly graphs. Graphs representing molecular structure have been used as an exemplar class due to their simplicity - the graphs are well constrained and have a standard nomenclature. The mechanisms developed for these graphs, however, demonstrate some principles which may be applied to the more general problem. Some of these have been discussed in work by other groups and merely applied here, but some are new.

1 The user must be in active control over the information flow, not just a passive reader. This was discussed by Stevens but is even more important in the case of graphs, perhaps to the extent that it is obvious - there is no clear way to just read a graph without providing control. Nevertheless it remains an important principle in the design of such systems.

2 There must be varying levels of detail available to the user: it should be possible to examine the structure of a graph at levels intermediate between the overall graph and its basic building blocks (the nodes and edges). This principle of providing overviews is similarly not a novel one. The mechanisms used for grouping nodes into collections for summary are different to those used for other types of information. In the instance of molecules the existence of recognisable groups which may be searched for and have proper names simplifies the matter considerably. Any program
designed to look at a generic graph would probably need a library of groups to search for (similar to the CML functional group files) for each intended application.

3 The procedures for finding structural features are more generally applicable, although the specific application will influence the priority given to certain features, e.g. if cycles are deemed to be particularly interesting features or not.

4 Transformation of the graph into a tree of overviews, as described above, can proceed automatically by measuring the complexity of the tree.

5 The navigation system has proven a useful mechanism for exploring the graph. The ability to navigate at a certain level then interrogate a particular group of nodes for more detail is an efficient method for building a mental image of the graph.

6 The provision of a GUI that is coordinated with speech should assist users with some vision, and will help teachers and colleagues.

The ideas developed during this work and outlined above should make useful guidelines for those who wish to deal with more general graphs. There is a great need for such systems, as described in the introduction to this thesis, and the applications are many - ER diagrams, UML, and flowcharts, to name but a few.

8.4 Conclusions

Despite the limited nature of this work, the limitations of the approach and areas not even investigated have been described above, it represents a progression in the field of assistive technologies. In a sense a complete system has been described - a piece of software has been developed that allows visually disabled users to conceptualise molecular structures - but its real contribution is as a prototype which demonstrates certain principles and provides a base upon which more sophisticated systems may be built.
Speech has been chosen as the non-visual output medium mainly due to its simplicity, cheapness and the fact that sound output is a standard feature on new computers. Furthermore no special skills or training are required to understand it. These are essential attributes if the software is to be used in schools. Speech is fully integrated with a graphical interface to assist users with some vision.

This work is a continuation of Stevens research into presentation of complex information, particularly mathematical equations. Stevens emphasised the importance of overviewing data before allowing its detailed exploration, with the user in control at all times. This work has placed these principles at the centre of its design and has demonstrated that they can be facilitated by transforming the graph into a tree-like data structure. Graph transformation is achieved by building a hierarchical tree of collections of nodes between each individual node at the bottom and the entire graph at the top. These collections can be described by a summary which indicates their nature without necessarily giving enough detail for a full understanding. In the case of chemical structures internationally recognised nomenclature provides an obvious, if not simple, means of summary.

Having built a tree of collections of nodes, conceptualisation of the graph is achieved by navigating around it. The user starts with a summary of the groups that comprise the top level (the entire graph) then investigates any of these that are not clear in more detail. This continues with the user unfolding the structure to a degree dependant on their prior knowledge of the molecule and their chemistry experience.

In conclusion this thesis has described the development of a system that enables visually disabled users to conceptualise molecular structures through speech. The principles developed have more general applicability and should provide useful guidelines for those wishing to design other tools for improving accessibility of complex information.
Glossary

This section contains a list of definitions for some terms used in this thesis. Please see sources \cite{32,33} for more information.

Chemistry Definitions

Atom: The smallest particle of an element.

Bond order: The number of electron pairs involved in covalent bonding between two atoms.

Covalent Bond: Chemical bond formed by the sharing of one or more electron pairs between two atoms.

Element: A substance that cannot be decomposed into simpler substances by chemical means.

Formula: Combination of symbols that indicates the chemical composition of a substance.

Heteroatom: Any atom in an organic molecule that is neither carbon nor hydrogen.


Molecule: The smallest particle of an element or compound capable of a stable, independent existence.

Organic: The chemistry of organic compounds; the chemistry of carbon compounds excluding the metal carbonates and the oxides and sulphides of carbon.
Other

CML: Chemical Markup Language. An application of XML.

DOM: Document Object Model. A programming interface developed by W3C allowing program objects to be built from XML documents.

DTD: Document Type Declaration. A specification that identifies how the markup is to be used.

ER: Entity relationship diagram. A data modelling technique that creates a graphical representation of the entities, and the relationship between them, in an information system.

Graph: A collection of nodes and edges.

GUI: Graphical User Interface.

JSAPI: Java Speech Application Programming Interface.

JSML: Java Speech Markup Language.

LEA: Local Education Authority.

Node: A point or vertex in a graph.

SSSR: Smallest Set of Smallest Rings.

Subgraph isomorphism: The existence of the nodes and edges of one graph in the same configuration as part of another graph.

Tree: A directed acyclic graph; i.e. a graph wherein there is only one route between any pair of nodes, and there is a notion of "toward top of the tree" (i.e. the root node), and its opposite direction, toward the leaves.
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML:</td>
<td>Extensible Markup Language. A recommendation of the world wide web consortium W3C allowing flexible representation of data.</td>
</tr>
</tbody>
</table>
Appendix 1  Organic Chemistry Functional Groups

A set of common functional groups found in organic chemistry are given below. There are many other groups which are named, especially more complex cyclic and polycyclic structures; these are generally built up from the groups below.

The atoms or bonds considered part of the functional group are drawn in blue. (R represents C or H)

**Hydrocarbon Groups:**

Alkane:

\[ R_4C \]

Alkene:

\[ R_2C=CR_2 \]

Alkyne:

\[ RC≡CR \]

Phenyl:

(Each carbon will have one other bond, in benzene all these are to a Hydrogen atom; it’s derivatives may have other groups. These are not shown.)

Alternatively represented as the kekulé structure:
Naphthalene:

Heteroatom groups:

Halide:
R-X \quad (X \text{ is any of the halogen atoms; } F, \text{ Cl, Br, I})

Hydroxyl:
R-OH \quad (\text{except if in a carboxylic acid})

Thiol:
R-SH

1° Amine:
R-NH_2

Nitrile:
R_3-C≡N

Isocyanate:
R-N=\text{C}=O

Nitro:
R-NO_2

Aldehyde:
Acid Halide:

\[
\begin{align*}
&\text{O} \\
&\text{C} \\
&\text{R} \\
&\text{X}
\end{align*}
\]

Carboxylic Acid:

\[
\begin{align*}
&\text{O} \\
&\text{C} \\
&\text{R'} \\
&\text{OH}
\end{align*}
\]

Sulphide:

\[\text{R-S-R}\]

Ether:

\[\text{R-O-R}\] (except if in an ester or acid anhydride)

Amine:

\[\text{R}_3\text{N}\] (1° amines are end groups)

Ketone:

\[
\begin{align*}
&\text{O} \\
&\text{C} \\
&\text{R} \\
&\text{R}
\end{align*}
\] (neither R is just H)

Acid Anhydride:

\[
\begin{align*}
&\text{O} \\
&\text{C} \\
&\text{R'} \\
&\text{O} \\
&\text{C} \\
&\text{R}
\end{align*}
\] (neither R is just H?)

Imine:

\[\text{R}_3\text{C}=\text{N-R}\]
Ester:

(neither R is just H)

Amide:

(end group if N is primary - i.e. R-C(O)NH₂)

Many other combinations exist, particularly complex and joined rings. Also not shown are some sulphur and phosphorous groups.
Appendix 2  Document Type Declaration for CML

<!-- CML V1.01 DTD -->
<!-- Amendments and errata from V1. (1999-05-15) -->
<!-- Authors:
P.Murray-Rust
H.Rzepa
This DTD is fully described in  P. Murray-Rust and H. S. Rzepa,
Journal of Chemical Information
and Computer Science,  1999, 39, 928.
-->  
<!-- Changes (2001-04-06):
1. atomRefs has been added as an attribute for string,
   float, integer and their *Arrays
2. some elements and attributes have been annotated as
   belonging to a subset "CoreCML" for simple small molecules
3. multiple dictRef on molecule removed
4. delimiter attribute added to floatArray, integerArray,
   floatMatrix
5. generic PE's created for many attributes for ease of maintenance
   no new attributes created
6. dictRef added to a very small number of elements - this is now
   almost universal
-->  
<!-- elements and attributes fall into 2 categories:
   coreCML - a subset of CML for easy implementation for small molecules
   fullCML - the full DTD (includes coreCML)
-->  
<!-- % fullCML can be overridden
by prepending the statement:
<!ENTITY % fullCML "IGNORE">
at this stage (or transmitting it in from the parser controls if
implemented) -->

<!ENTITY % coreCML "INCLUDE">
<!ENTITY % fullCML "INCLUDE">

<!-- PARAMETER ENTITIES
---

--- attributes found on almost all elements
---

<!ENTITY % title 'title CDATA #IMPLIED'>
<!ENTITY % id 'id CDATA #IMPLIED'>
<!ENTITY % convention 'convention CDATA "CML"'>
<!ENTITY % dictRef 'dictRef CDATA #IMPLIED'>

<!-- a very common combination (on most elements) -->
<!ENTITY % tit_id_conv '%$title; %id; %convention;'>
<!ENTITY % tit_id_conv_dict '%$tit_id_conv; %dictRef;'>

<!-- linking information
---

<!ENTITY % simpleLink 'href CDATA #REQUIRED'>

<!-- quantifiers and constraints on some primitives
---

97
<!ENTITY % count "count CDATA "1"">
<!ENTITY % size "size CDATA #IMPLIED">
<!ENTITY % rows "rows CDATA #REQUIRED">
<!ENTITY % columns "columns CDATA #REQUIRED">]
<!ENTITY % size ''>  
<!ENTITY % rows ''>  
<!ENTITY % columns ''>  

<!-- ====== delimiter in Array elements ======-->  
<!ENTITY % delimiter "delimiter CDATA #IMPLIED">  

<!-- ====== constraints on some numeric data primitives ======-->  
<!ENTITY % units "units CDATA #IMPLIED">  
<!ENTITY % min "min CDATA #IMPLIED">  
<!ENTITY % max "max CDATA #IMPLIED">  
<!ENTITY % angleUnits "units (degrees | radians) "degrees"">  
<!ENTITY % unitsRef "unitsRef CDATA #IMPLIED">  

<!-- for CoreCML degrees are mandatory -->  
<!ENTITY % angleUnits "units CDATA #FIXED "degrees"">  

<!-- values which may be useful in min and max attributes -->  
<!ENTITY % integer.zero '0'>  
<!ENTITY % integer.max '2147483647'>  
<!ENTITY % integer.min '-2147483648'>  
<!ENTITY % float.zero '0.0'>  
<!ENTITY % float.max '8.98846567431158E307'>  
<!ENTITY % float.min '4.9E-324'>  

<!-- ====== builtin values for any element ============-->  
<!ENTITY % builtinId 'id'>  

<!-- ====== builtin values for atoms ============-->  
<!ENTITY % elementType 'elementType'>  

<![ %fullCML; [  
<!ENTITY % atomId 'atomId '>
]]>  
<!ENTITY % atomId ''>  

<!ENTITY % x2 'x2'>  
<!ENTITY % y2 'y2'>  
<!ENTITY % x3 'x3'>  
<!ENTITY % y3 'y3'>  
<!ENTITY % z3 'z3'>  
<!ENTITY % xy2 'xy2'>  
<!ENTITY % xyz3 'xyz3'>  
<!ENTITY % xFrat 'xFrat'>  
<!ENTITY % yFrat 'yFrat'>  
<!ENTITY % zFrat 'zFrat'>  
<!ENTITY % xyzFrat 'xyzFrat'>
<!ENTITY % occupancy 'occupancy'>
<!ENTITY % isotope 'isotope'>
<!ENTITY % formalCharge 'formalCharge'>
<!ENTITY % nonHydrogenCount 'nonHydrogenCount'>
<!ENTITY % hydrogenCount 'hydrogenCount'>
<!ENTITY % atomParity 'atomParity'>

<![%fullCML;[
<!ENTITY % residueType ' | residueType'>
<!ENTITY % residueId ' | residueId'>
]]>
<!ENTITY % residueType ''>
<!ENTITY % residueId ''>

<!ENTITY % atomStringBuiltin '%elementType; %atomId; %residueType; %residueId; '

<!ENTITY % atomFloatBuiltin '%x2; | %y2; | %x3; | %y3; | %z3; | %xFract; | %yFract; | %zFract; | %occupancy; | %isotope; | %formalCharge; | %hydrogenCount; | %nonHydrogenCount; | %atomParity; '

<!ENTITY % atomIntegerBuiltin '%isotope; | %formalCharge; | %hydrogenCount; | %nonHydrogenCount; | %atomParity; '

<!ENTITY % atomCoordinate2Builtin '%xy2; '

<!ENTITY % atomCoordinate3Builtin '%xyz3; | %xyzFract; '

<!--- ========= builtin values for bonds ================>
<!ENTITY % atomRef 'atomRef'>
<!ENTITY % builtinAtomRefs 'atomRefs'>
<!ENTITY % length 'length'>
<!ENTITY % order 'order'>
<!ENTITY % stereo 'stereo'>
<!ENTITY % atomRefs 'atomRefs CDATA #IMPLIED'>

<!ENTITY % bondStringBuiltin 'DATA #IMPLIED'>
<!ENTITY % atomRef; | %builtinAtomRefs; | %order; | %stereo; '

<!ENTITY % bondFloatBuiltin '%length; '

<!ENTITY % bondIntegerBuiltin ''
<!--- ====== builtin values for crystal ====================>
<!ENTITY % acell   'acell'>
<!ENTITY % bcell   'bcell'>
<!ENTITY % ccell   'ccell'>
<!ENTITY % alpha   'alpha'>
<!ENTITY % beta   'beta'>
<!ENTITY % gamma   'gamma'>
<!ENTITY % z       'z'>
<!ENTITY % spacegroup 'spacegroup'>

<!ENTITY % crystalStringBuiltin ' %spacegroup; '>
<!ENTITY % crystalFloatBuiltin ' %acell; | %bcell; | %ccell; |
   %alpha; | %beta; | %gamma; |
   %z; '>

<!ENTITY % crystalIntegerBuiltin ' %z; '>

<!--- ================================================>
<!ENTITY % stringBuiltin ' builtin ( %builtinId; | %atomStringBuiltin; |
   %bondStringBuiltin; | %crystalStringBuiltin;
 ) #IMPLIED '>

<!ENTITY % floatBuiltin ' builtin ( %atomFloatBuiltin; | %bondFloatBuiltin; |
   %crystalFloatBuiltin;
 ) #IMPLIED '>

<!ENTITY % integerBuiltin ' builtin ( %atomIntegerBuiltin; | %crystalIntegerBuiltin;
 ) #IMPLIED '>

<!ENTITY % coordinate2Builtin ' builtin ( %atomCoordinate2Builtin;
 ) #IMPLIED '>

<!ENTITY % coordinate3Builtin ' builtin ( %atomCoordinate3Builtin;
 ) #IMPLIED '>

<!--- ================================================>
<!ENTITY % ELEMENTS for widely used data primitives -->
<!-- CML-DTD V1.01 addition
 certain children of atom or bond may refer to atoms (particularly
 builtin="atomParity" and builtin="stereo"). For these
 we need the attribute 'atomRefs'
 -->
<!ENTITY % scalar.content  '#PCDATA'>
<!ENTITY % array.content  '#PCDATA'>
<!ENTITY % matrix.content  '#PCDATA'>
<!ENTITY % angle.content  '#PCDATA'>
<!ENTITY % coordinate.content  '#PCDATA'>
<!ENTITY % array.atts
 'size;
 delimiter;'
>
<!ELEMENT string  %scalar.content;>
<!ATTLIST string
 %tit_id_conv_dict;
 %stringBuiltin;
 %atomRefs;
>
<!ELEMENT float  %scalar.content;>
<!ATTLIST float
 %tit_id_conv_dict;
 %floatBuiltin;
 %min;
 %max;
 %units;
 %unitsRef;
 %atomRefs;
>
<!ELEMENT integer  %scalar.content;>
<!ATTLIST integer
 %tit_id_conv_dict;
 %integerBuiltin;
 %min;
 %max;
 %units;
 %unitsRef;
 %atomRefs;
>
<!ELEMENT stringArray  %array.content;>
<!ATTLIST stringArray
 %tit_id_conv_dict;
 %stringBuiltin;
 %array.atts;
 %min;
 %max;
Appendix 3  Test Molecules

The following molecules are those selected to test the ideas developed during this work and their implementation. They are intended to represent a range of 'styles' of molecule from simple to relatively complex. Some of them may not, or may not be able to exist, in nature, but have been designed purely for testing purposes.

The CML describing these molecules is given after the descriptions.

Molecule 1.

Ethanol.

<?xml version="1.0"?>
<!DOCTYPE molecule SYSTEM "cml.dtd" []>
<molecule name="ethanol">
  <atomArray>
    <atom id="a1">
      <string builtin="elementType">O</string>
      <float builtin="x2">10.7667</float>
      <float builtin="y2">-5.6873</float>
    </atom>
    <atom id="a2">
      <string builtin="elementType">C</string>
      <float builtin="x2">6.8780</float>
      <float builtin="y2">-7.7773</float>
    </atom>
    <atom id="a3">
      <string builtin="elementType">C</string>
      <float builtin="x2">9.2720</float>
      <float builtin="y2">-7.7773</float>
    </atom>
  </atomArray>
  <bondArray>
    <bond id="b1">
      <string builtin="atomRef">a3</string>
      <string builtin="atomRef">a2</string>
      <string builtin="order">1</string>
    </bond>
    <bond id="b2">
      <string builtin="atomRef">a1</string>
      <string builtin="atomRef">a3</string>
      <string builtin="order">1</string>
    </bond>
  </bondArray>
</molecule>
Molecule 2.

Hexanamide.

<?xml version="1.0"?>
<!DOCTYPE molecule SYSTEM "cml.dtd" []>
<molecule name="haxanamide">
<atomArray>
  <atom id="a1">
    <string builtin="elementType">C</string>
    <float builtin="x2">8.1702</float>
    <float builtin="y2">-7.8136</float>
  </atom>
  <atom id="a2">
    <string builtin="elementType">C</string>
    <float builtin="x2">9.5138</float>
    <float builtin="y2">-7.0379</float>
  </atom>
  <atom id="a3">
    <string builtin="elementType">C</string>
    <float builtin="x2">10.8574</float>
    <float builtin="y2">-7.8136</float>
  </atom>
  <atom id="a4">
    <string builtin="elementType">C</string>
    <float builtin="x2">12.2010</float>
    <float builtin="y2">-7.0379</float>
  </atom>
  <atom id="a5">
    <string builtin="elementType">C</string>
    <float builtin="x2">13.5446</float>
    <float builtin="y2">-7.8136</float>
  </atom>
  <atom id="a6">
    <string builtin="elementType">C</string>
    <float builtin="x2">14.8882</float>
    <float builtin="y2">-7.0379</float>
  </atom>
  <atom id="a7">
    <string builtin="elementType">O</string>
    <float builtin="x2">14.8882</float>
    <float builtin="y2">-5.4864</float>
  </atom>
  <atom id="a8">
    <string builtin="elementType">N</string>
    <float builtin="x2">16.2318</float>
    <float builtin="y2">-7.8136</float>
  </atom>
</atomArray>
Molecule 3.

(2E)-3-phenylpenta-2,4-dienoic acid.
<?xml version="1.0"?>
<!DOCTYPE molecule SYSTEM "cml.dtd" []>
<molecule name="(2E)-3-phenylpenta-2,4-dienoic acid">
<atomArray>
  <atom id="a1">
    <string builtin="elementType">O</string>
    <float builtin="x2">13.5621</float>
    <float builtin="y2">-4.1826</float>
  </atom>
  <atom id="a2">
    <string builtin="elementType">C</string>
    <float builtin="x2">8.2552</float>
    <float builtin="y2">-6.0210</float>
  </atom>
  <atom id="a3">
    <string builtin="elementType">C</string>
    <float builtin="x2">10.3779</float>
    <float builtin="y2">-6.0210</float>
  </atom>
  <atom id="a4">
    <string builtin="elementType">C</string>
    <float builtin="x2">11.4393</float>
    <float builtin="y2">-7.8593</float>
  </atom>
  <atom id="a5">
    <string builtin="elementType">C</string>
    <float builtin="x2">13.5621</float>
    <float builtin="y2">-7.8593</float>
  </atom>
  <atom id="a6">
    <string builtin="elementType">C</string>
    <float builtin="x2">14.6234</float>
    <float builtin="y2">-6.0210</float>
  </atom>
  <atom id="a7">
    <string builtin="elementType">O</string>
    <float builtin="x2">16.7462</float>
    <float builtin="y2">-6.0210</float>
  </atom>
  <atom id="a8">
    <string builtin="elementType">C</string>
    <float builtin="x2">10.3779</float>
    <float builtin="y2">-9.6977</float>
  </atom>
  <atom id="a9">
    <string builtin="elementType">C</string>
    <float builtin="x2">11.4393</float>
    <float builtin="y2">-11.5361</float>
  </atom>
  <atom id="a10">
    <string builtin="elementType">C</string>
    <float builtin="x2">8.2552</float>
    <float builtin="y2">-9.6977</float>
  </atom>
  <atom id="a11">
    <string builtin="elementType">C</string>
    <float builtin="x2">10.3779</float>
    <float builtin="y2">-13.3744</float>
  </atom>
  <atom id="a12">
    <string builtin="elementType">C</string>
Molecule 4.

2-amino-3-phenylpropanoic acid (phenylalanine).

<?xml version="1.0"?>
<!DOCTYPE molecule SYSTEM "cml.dtd" []>
<molecule name="phenylalanine">
<atomArray>
<atom id="a1">
<string builtin="elementType">O</string>
<float builtin="x2">11.0216</float>
<float builtin="y2">-5.8519</float>
</atom>
<atom id="a2">
<string builtin="elementType">C</string>
<float builtin="x2">11.0216</float>
<float builtin="y2">-7.1820</float>
</atom>
<atom id="a3">
<string builtin="elementType">C</string>
<float builtin="x2">12.1735</float>
<float builtin="y2">-7.8470</float>
</atom>
<atom id="a4">
<string builtin="elementType">O</string>
<float builtin="x2">9.8697</float>
<float builtin="y2">-7.8470</float>
</atom>
<atom id="a5">
<string builtin="elementType">C</string>
</atom>
</atomArray>
</molecule>
<atom>
  <float builtin="x2">13.3253</float>
  <float builtin="y2">-7.1820</float>
</atom>

<atom id="a6">
  <string builtin="elementType">N</string>
  <float builtin="x2">12.1735</float>
  <float builtin="y2">-9.1771</float>
</atom>

<atom id="a7">
  <string builtin="elementType">C</string>
  <float builtin="x2">14.4772</float>
  <float builtin="y2">-7.8470</float>
</atom>

<atom id="a8">
  <string builtin="elementType">C</string>
  <float builtin="x2">15.6291</float>
  <float builtin="y2">-7.1820</float>
</atom>

<atom id="a9">
  <string builtin="elementType">C</string>
  <float builtin="x2">14.4772</float>
  <float builtin="y2">-9.1771</float>
</atom>

<atom id="a10">
  <string builtin="elementType">C</string>
  <float builtin="x2">15.6291</float>
  <float builtin="y2">-9.8421</float>
</atom>

<atom id="a11">
  <string builtin="elementType">C</string>
  <float builtin="x2">16.7810</float>
  <float builtin="y2">-7.8470</float>
</atom>

<atom id="a12">
  <string builtin="elementType">C</string>
  <float builtin="x2">16.7810</float>
  <float builtin="y2">-9.1771</float>
</atom>

</atomArray>

<bondArray>
  <bond id="b1">
    <string builtin="atomRef">a3</string>
    <string builtin="atomRef">a2</string>
    <string builtin="order">1</string>
  </bond>

  <bond id="b2">
    <string builtin="atomRef">a4</string>
    <string builtin="atomRef">a2</string>
    <string builtin="order">2</string>
  </bond>

  <bond id="b3">
    <string builtin="atomRef">a5</string>
    <string builtin="atomRef">a3</string>
    <string builtin="order">1</string>
  </bond>

  <bond id="b4">
    <string builtin="atomRef">a6</string>
    <string builtin="atomRef">a3</string>
    <string builtin="order">1</string>
  </bond>

  <bond id="b5">
Molecule 5.

Tetrahydro-2\textit{H}-pyran-2-one.
Molecule 6.

1-chloro-3,4-dihydro-1$H$-isochromene-8-carbonitrile.

<?xml version="1.0"?>
<!DOCTYPE molecule SYSTEM "cml.dtd" []>
<molecule name="1-chloro-3,4-dihydro-1H-isochromene-8-carbonitrile">
<atomArray>
<atom id="a1">
<string builtin="elementType">N</string>
<float builtin="x2">12.7371</float>
<float builtin="y2">-12.2503</float>
</atom>
<atom id="a2">
<string builtin="elementType">C</string>
<float builtin="x2">13.8890</float>
<float builtin="y2">-11.5853</float>
</atom>
<atom id="a3">
<string builtin="elementType">Cl</string>
<float builtin="x2">12.7371</float>
<float builtin="y2">-9.5902</float>
</atom>
<atom id="a4">
<string builtin="elementType">O</string>
</atom>
</atomArray>
</molecule>
<atom id="a5">
  <string builtin="elementType">C</string>
  <float builtin="x2">15.0409</float>
  <float builtin="y2">-6.9300</float>
</atom>
<atom id="a6">
  <string builtin="elementType">C</string>
  <float builtin="x2">16.1928</float>
  <float builtin="y2">-7.5951</float>
</atom>
<atom id="a7">
  <string builtin="elementType">C</string>
  <float builtin="x2">17.3446</float>
  <float builtin="y2">-9.5902</float>
</atom>
<atom id="a8">
  <string builtin="elementType">C</string>
  <float builtin="x2">17.3446</float>
  <float builtin="y2">-10.9203</float>
</atom>
<atom id="a9">
  <string builtin="elementType">C</string>
  <float builtin="x2">16.1928</float>
  <float builtin="y2">-8.9251</float>
</atom>
<atom id="a10">
  <string builtin="elementType">C</string>
  <float builtin="x2">16.1928</float>
  <float builtin="y2">-11.5853</float>
</atom>
<atom id="a11">
  <string builtin="elementType">C</string>
  <float builtin="x2">15.0409</float>
  <float builtin="y2">-9.5902</float>
</atom>
<atom id="a12">
  <string builtin="elementType">C</string>
  <float builtin="x2">15.0409</float>
  <float builtin="y2">-10.9203</float>
</atom>
<atom id="a13">
  <string builtin="elementType">C</string>
  <float builtin="x2">13.8890</float>
  <float builtin="y2">-8.9251</float>
</atom>
</atomArray>
<bondArray>
<bond id="b1">
  <string builtin="atomRef">a8</string>
  <string builtin="atomRef">a7</string>
  <string builtin="order">1</string>
</bond>
<bond id="b2">
  <string builtin="atomRef">a9</string>
  <string builtin="atomRef">a7</string>
  <string builtin="order">2</string>
</bond>
<bond id="b3">
References

1 <http://www.rnib.org.uk/wesupply/fctsheet/authuk.htm>

2 Bruce, I. et al; Blind and partially sighted adults in Britain: the RNIB survey. HMSO. 1991.


10 Lange, M.O.; Tactile Graphics - as easy as that! In: *Proceedings of the CSUN Technology and Persons with Disabilities Conference 1999*.


For example ACD/Chemsketch\cite{ACD} or the JME (Java Molecular Editor) available from the CML website\cite{cml}. 


\cite{cml}

<http://www.xml-cml.org>


33 Computing related definitions from whatis.com <http://www.whatis.techtarget.com/>