Automated Analysis and Validation of Chemical Literature

Joseph Andrew Townsend

Corpus Christi College

A dissertation submitted to the University of Cambridge
for the degree of Doctor of Philosophy

Unilever Centre for Molecular Science Informatics
Department of Chemistry
Lensfield Road,
Cambridge, CB2 1EW,
United Kingdom.

August 5, 2007

Disclaimer

This thesis is the result of my own work and includes nothing which is the outcome of work done in collaboration, except where specifically indicated.

This thesis does not exceed the specified word limit (60000) as defined by the Chemistry Degree Committee.

This thesis has been typeset in 12pt font using $\LaTeX 2\varepsilon$ according to the specifications defined by the Board of Graduate Studies and the Chemistry Degree Committee.

Abstract

Methods to automatically extract and validate data from the chemical literature in legacy formats to machine-understandable forms are examined. The works focuses of three types of data: analytical data reported in articles, computational chemistry output files and crystallographic information files (CIFs).

It is shown that machines are capable of reading and extracting analytical data from the current legacy formats with high recall and precision. Regular expressions cannot identify chemical names with high precision or recall but non-deterministic methods perform significantly better. The lack of machine-understandable connection tables in the literature has been identified as the major issue preventing molecule-based data-driven science being performed in the area.

The extraction of data from computational chemistry output files using parser-like approaches is shown to be not generally possible although such methods work well for input files. A hierarchical regular expression based approach can parse $> 99.9\%$ of the output files correctly although significant human input is required to prepare the templates. CIFs may be parsed with extremely high recall and precision, contain connection tables and the data is of high quality.

The comparison of bond lengths calculated by two computational chemistry programs show good agreement in general but structures containing specific moieties cause discrepancies. An initial protocol for the high-throughput geometry optimisation of molecules extracted from the CIFs is presented and the refinement of this protocol is discussed. Differences in bond length between calculated and experimentally determined values from the CIFs of less than 0.03Å are shown to be expected by random error. The final protocol is used to find high-quality structures from crystallography which can be reused for further science.

Acknowledgments

I would like to thank Dr Jonathan Goodman, Dr Peter Murray-Rust, Sam Adams, Fraser Norton and Chris Waudby for their help and advice on the OSCAR project. Dr Murray-Rust, Dr Simon "Billy" Tyrrell, Dr Yong "YY" Zhang and Nick Day are also thanked for their support and cooperation during the course of my PhD. Dr Charlotte Bolton deserves special mention for providing the computational support without which this work would not have been possible. The Pickerel and Helen Clubb also deserve mention for allowing me to sanity check my ideas and for general support. The Unilever Centre for Molecular Science Informatics and the Royal Society of Chemistry are thanked for funding.

Contents

List of Tables

List of Figures

Glossary

BNF	Backus-Naur form
CAS	Chemical Abstracts Service
CCDC	Cambridge Crystallographic Data Centre
CIF	Crystallographic Information File
CML	Chemical Markup Language
CNMR	Carbon Nuclear Magnetic Resonance
CSD	Cambridge Structural Database
CSS	Cascading Style Sheet
CT	Connection Table
DFT	Density Functional Theory
DOM	Document Object Model
DTD	Document Type Definition
EDC	Experimental Data Checker
GTO	Gaussian Type Orbital
HT	High-Throughput
HNMR	Proton Nuclear Magnetic Resonance
HRMS	High Resolution Mass Spectroscopy
HTML	Hyper Text Markup Language
IAM	Independent Atom Model
IE	Information Extraction
IUCr	International Union of Crystallography
IUPAC	International Union of Pure and Applied Chemistry
$\mathbf{m}_u$	Unified Atomic Mass Unit 1.6605×10^{-27} kg
MP2	Møller-Plesset perturbation theory, second order

NCI National Cancer Institute

NMR Nuclear Magnetic Resonance

OBC Organic & Biomolecular Chemistry

OOP Object-Orientated Programming

OSCAR Open Source Chemistry Analysis Routines

QQ Quantile-Quantile

RSC Royal Society of Chemistry

SCF Self Consistent Field

SGML Standard Generalised Markup Language

STO Slater Type Orbital

SVG Scalable Vector Graphics

UCC Unilever Centre for Molecular Science Informatics

W3C World Wide Web Consortium

WS WebService

XML eXtensible Markup Language

XSL eXtensible Style Language

XSLT eXtensible Style Language Transformations

$\overline{x}$ The arithmetic mean of the data type x, given by

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{1}$$

s The standard deviation of a sample of n instances of data type x, given by

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2} \tag{2}$$

ρ Spearman's rank correlation coefficient, given by

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)} \tag{3}$$

where d_i is the difference between each rank of corresponding values of x and y

W The test statistic for the Shapiro-Wilk normality test given by

$$W = \frac{\left(\sum_{i=1}^{n} a_i x_{(i)}\right)^2}{\sum_{i=1}^{n} \left(x_i - \overline{x}\right)^2} \tag{4}$$

where $x_{(i)}$ is the i^{th} order statistic, *i.e.* the i^{th}-smallest number in the sample and the constants a_i are given by

$$(a_i, \ldots, a_n) = \frac{m^T V^{-1}}{\sqrt{(m^T V^{-1} V^{-1} m)}} \tag{5}$$

where

$$m = (m_1, \ldots, m_n)^T \tag{6}$$

and m_i are the expected values of the order statistics of independent and identically-distributed random variables sampled from the standard normal distribution and V is the covariance matrix of those order statistics.

Chapter 1

Introduction

The volume of scientific data* being produced is exploding and phrases such as *data deluge*, *data avalanche* and *digital deluge* are now commonly seen in the literature [1]. As an illustration of the scale of the current problem Lesk estimates that the Bible requires 5×10^6 bytes of storage [2] whereas the annual production of refereed journal literature is estimated by Hey *et al.* to require 1×10^{12} bytes [1]. The volume of data produced in the fields of astronomy and bioscience are doubling in 12 and 9 months respectively [3]. This is faster than the measured rate of increase in performance of computer chips (Moore's Law), which doubles every 18 months. However, modern analysis methods are also much faster which should result in an ideal situation.

The information explosion is causing problems for many scientific communities because the data production and data analysis processes are currently disconnected [4]. That is, the data is not available in suitable formats for re-use, often as a result of the publication process. Data is seen as an integral part of any scientific discipline and the ability to view and analyse another scientist's data is essential. The current reliance on paper publishing prevents much of the data produced during an experiment ever being published. It is now common to see phrases such as

...one hundred and five optimised geometries had been obtained

*The word data is used frequently in this thesis as a synonym for a collection of information and as the plural of datum — the meaning should be apparent from context.

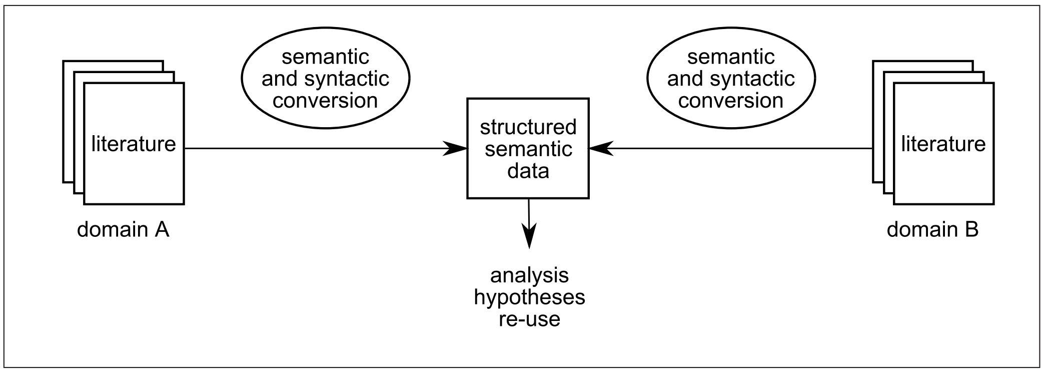
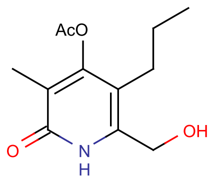


Figure 1.1: Data from different domains can be extracted (and combined) to a more usable form.

> in the present work. Reporting all of these optimised geometries is nonsense [5]

in an article and it is estimated that ca. 80% of crystal structures determined are never published [6, 7]. The publication of thesis present similar problems — approximately 60 gigabytes of analysis, programs and most importantly data were created as a part of this thesis. It would be possible to include this as supplementary data by requesting that a CD is included as part of the thesis — although to include all the data would actually require nearly 100 CDs [8].

The publication process forces the decoupling of the interpretation and analysis (which is published) from data (which often is not published in full). There is a desire amongst the community to improve this situation, enabling more rapid publication and dissemination processes. In future, it is hoped that everything can be captured and published [10]; Hagdorn's recent thesis is an example of how theses may appear in the future [9]. However, solving this problem will increase the information overload.



4-Acetoxy-6-hydroxymethyl-3-methyl-5-propyl-1H-pyridin-2-one (26). $NaBH_4$ (0.127 g, 3.36 mmol) was added to a solution of aldehyde **24** (0.662 g, 2.79 mmol) in EtOH (18 mL) at 0 °C and the mixture was stirred for 2.5 h at rt. After addition of sat. NH_4Cl solution (10 mL), the mixture was concentrated in vacuo and then extracted with EtOAc (5 × 25 mL). The combined organic phases were dried over Na_2SO_4, then filtered and concentrated in vacuo. The residue was recrystalized from EtOAc to yield 0.58 g (87 %) of alcohol **26** as colorless crystals. — ^{1}H NMR (250 MHz, $CDCl_3$): δ 0.92 (t, ^3J = 7.2 Hz, 3 H), 1.41 (m_c, 2 H), 1.91 (s, 3 H), 2.22 (t, ^3J = 7.4 Hz, 2 H), 2.34 (s, 3 H), 4.65 (s, 2 H).

Figure 1.2: A typical paragraph from the supplemental data of an article in Journal of Organic Chemistry [14].

1.1 Data-Driven Science

To allow data-driven science (and further re-use), methods to extract data in legacy formats to more usable forms are required and the sheer volume of data requires that any process to do this must be automated. This work considers possible approaches for the creation of such transformation processes and their applicability with the premise that

> . . . the current scientific literature, were it to be presented in semantically accessible form, contains huge amounts of undiscovered science. [11]

The data deluge is not confined to any one scientific domain therefore multiple tools may be required to extract it to a reusable form (figure 1.1).

A human is clearly not capable of performing an analysis of all the annual journal literature unaided, or even just that of the chemical literature — Chemical Abstracts Service reported over 1 million articles in 2006 and now contains the abstracts of over 25 million articles [12, 13]. A typical article in the Journal of Organic Chemistry (ca. 8 pages) frequently has supporting supplemental data (which may not be refereed). The volume of the supplementary data is often five to ten times that of the original article, further exacerbating the problem. However, once the data is in the relevant form a computer can be used to help analyse it, allowing humans to interpret abnormalities, trends and other features of interest. This is a prime motivation for representing the data in a *machine-understandable*[†] form.

1.2 Metadata, Syntax and Semantics

Metadata[‡] is essential for both the automation of the extraction process and the subsequent machine-understandability. Metadata such as those developed by the Dublin Core Metadata Initiative allow the providence and reusability of data to be interpreted by a machine [16].

For extraction to be possible, the *semantics* and *syntax* of the legacy format must be understood — semantics refer to the meaning and syntax to the structure of the data. The chemical field (especially organic chemistry) contains-well understood semantics and syntax which have been stable for a long time and can be represented in a machine-understandable format using Chemistry Markup Language. There is also a large legacy corpus. As a result organic chemistry represents an attractive area for data extraction.

The most semantically and syntactically rich area of organic chemistry is the synthesis of a molecule and the associated analytical data; an example of such data is shown in figure 1.2. The legacy corpus consists of two major document types: journal articles and theses. The structure of the two types differs but the data is expected to be recoverable using natural language techniques (figure 1.3). Whilst significant progress was made in this area —

[†]The term machine-understandable is explained in section 1.5

[‡]Metadata is commonly defined as *data that is used to describe other data*, this might range from who authored the data to the fact that the document conforms to a particular DTD [15].

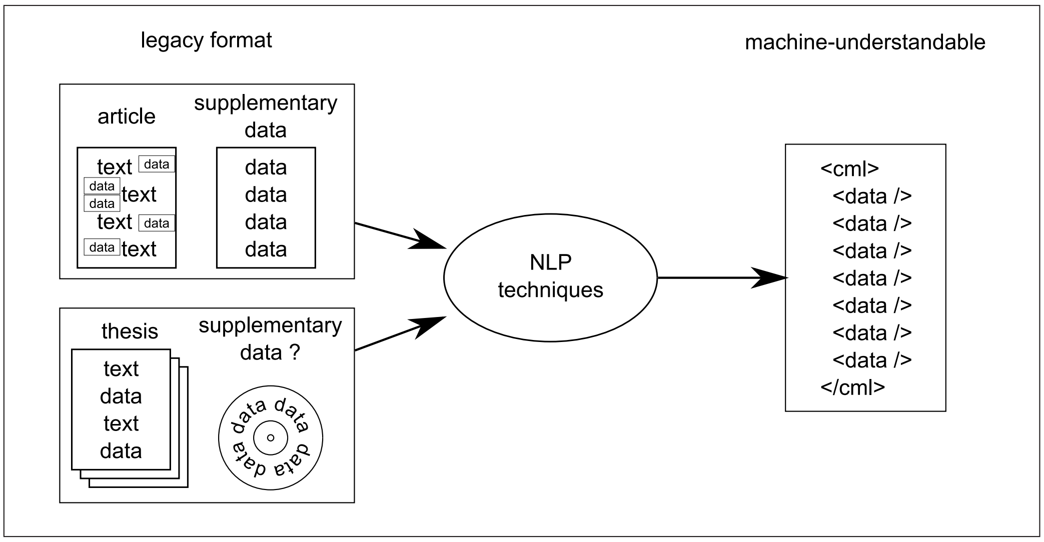


Figure 1.3: The general structure of journal articles and theses differ but should both be amenable to natural language parsing techniques.

including the development of ideas applicable to other scientific fields — the extraction did not prove to be as tractable as expected.

1.3 Data Validation

Chemistry is based on the synthesis, structure and properties of molecules; it is therefore vital that a molecule can be uniquely and globally identified. IUPAC's International Chemical Identifier (InChI) provides an ideal solution [17]. There are often multiple instances of a particular molecule and properties[§] associated with it reported in the literature. By using unique identifiers and appropriate metadata all the instances can be combined to give an overall picture (figure 1.4).

In general, the *ideal value* of a property associated with a molecule can never be known. However, properties can be observed or calculated and the agreement between multiple instances of this property reinforces the belief

[§]A property in this case refers to physical quantities relating to that molecule, such as the melting point or chemical formula.

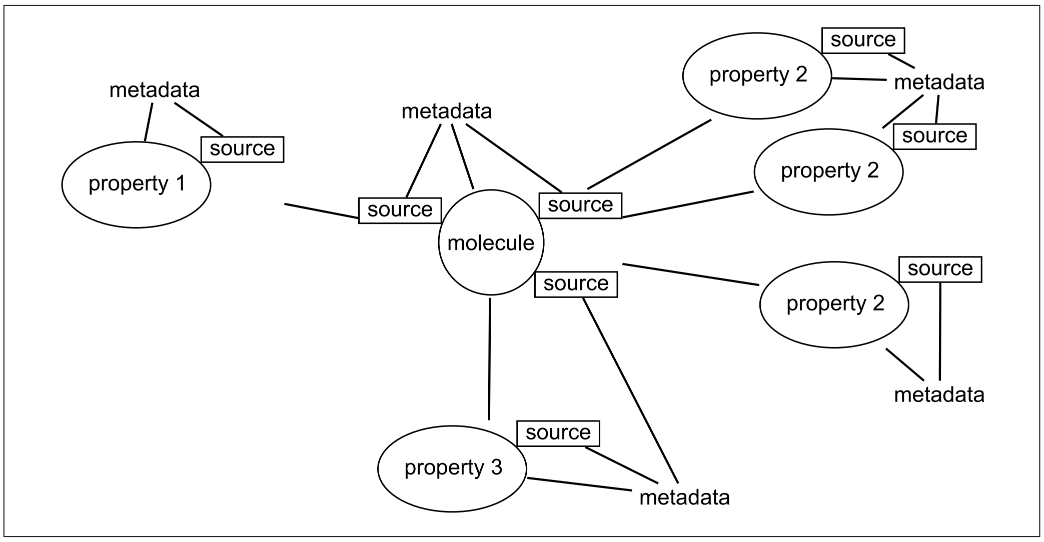


Figure 1.4: A molecule may have multiple properties associated with it; repeated instances of a particular property may be reported independently.

that the ideal value is being approached. Conversely disagreement between the values can lead to improvements of the measurement method or calculation (figure 1.5). Stewart found significant errors in the literature pertaining to experimentally-determined enthalpies of formation which were detected by comparison with calculated values [18]. In his own words

> . . . the accuracy of experimental enthalpies of formation can be investigated by using semiempirical methods. Where agreement between calculated and reported enthalpies of formation exists, it is reasonable to assume that the experimental value is, indeed, accurate. Where there is a large difference, the error is likely to be either in the computational method or in the experiment; the probability of both being equally incorrect is small. [18]

More recently, the structure of hexacyclinol (figure 1.6) has been the matter of debate. Gräfe proposed a structure for the molecule in 2002 [19]. Following a total synthesis, La Clair confirmed this structure [20]. Rych-

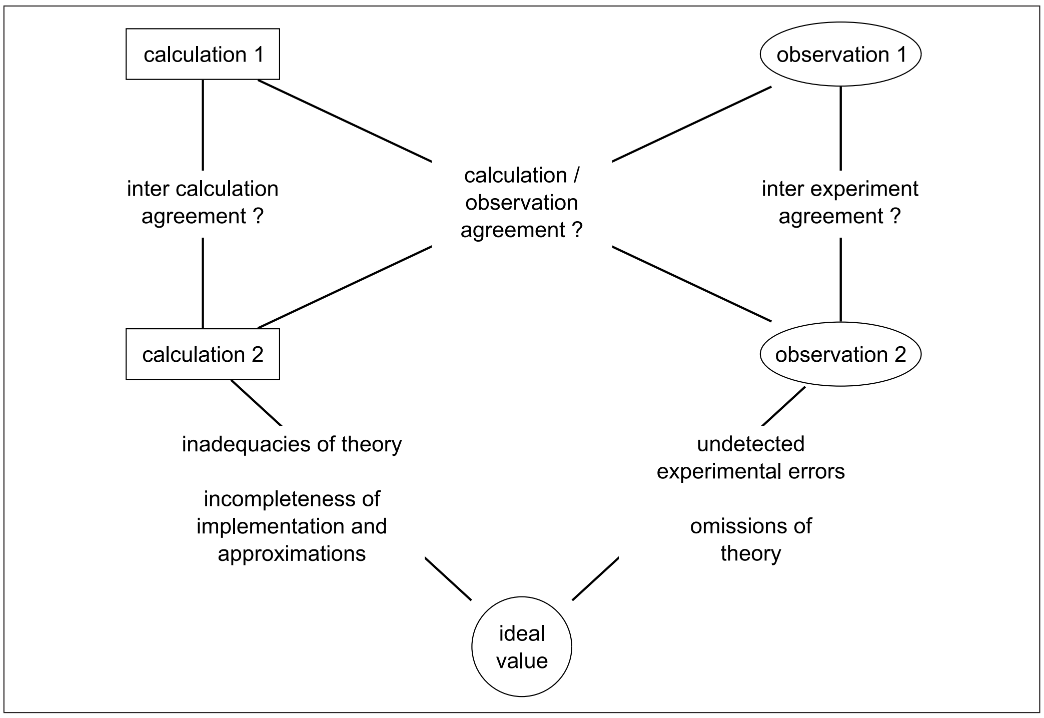

calculation 1
observation 1
inter calculation agreement ?
calculation / observation agreement ?
inter experiment agreement ?
calculation 2
observation 2
inadequacies of theory
incompleteness of implementation and approximations
undetected experimental errors
omissions of theory
ideal value

Figure 1.5: The agreement between different instances of a property can reinforce the belief that the value is approaching the ideal value. Conversely, differences between the values can lead to refinement of the calculation or experimental processes so a more accurate value can be found.

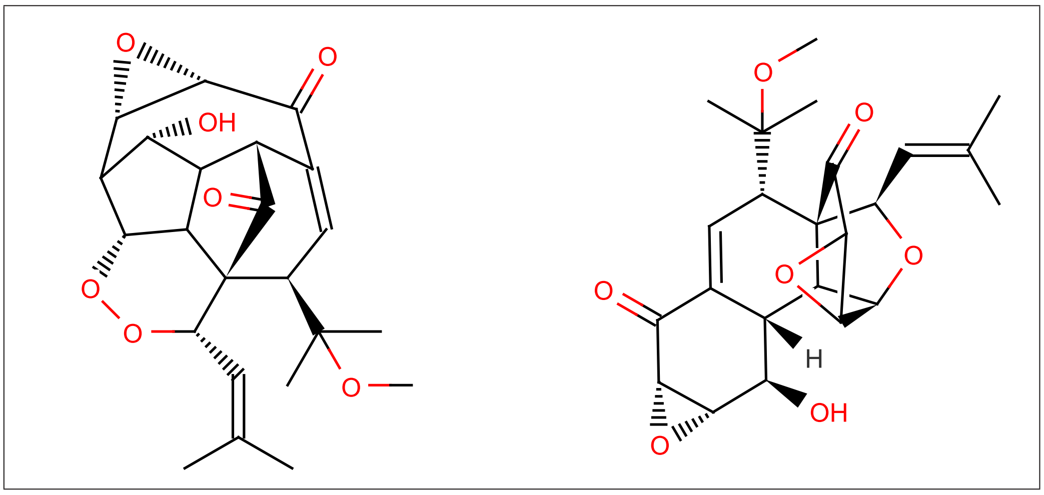


Figure 1.6: The structures of hexacyclinol proposed by Gräfe (left) and Rychnovsky (right).

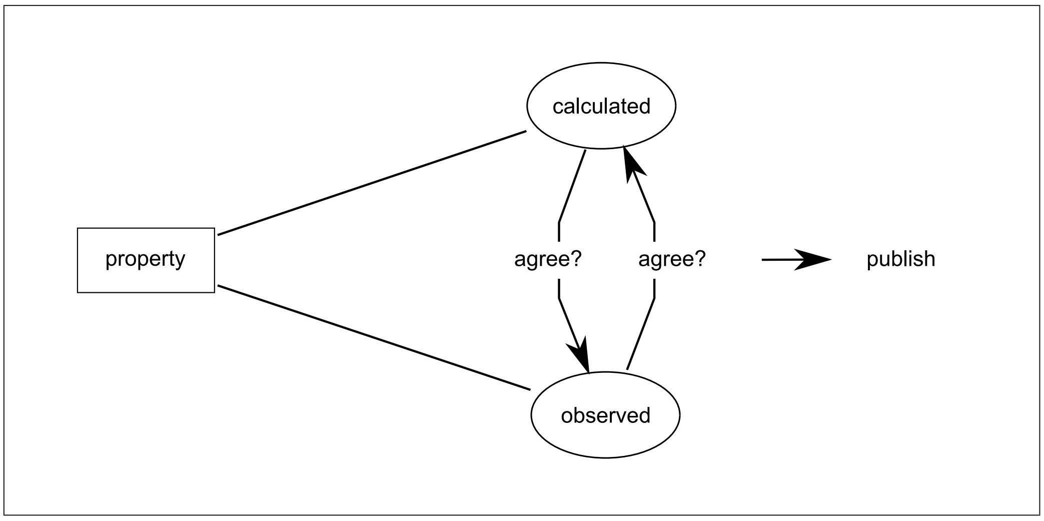


Figure 1.7: The increased availability of computational resources will hopefully lead to the validation of both calculated and observed properties prior to publication.

novsky simulated the compound's CNMR spectrum to see if it agreed with that reported and came to the conclusion that

> ...the original structure assignment doesn't fit the data. [21]

Using the same computational method, a revised structure was proposed [21]. It is expected that the increased availability and speed of computing resources will in future allow such a validation to become a matter of course, leading to the situation shown in figure 1.7. For such a situation to arise requires the computational chemistry output to be machine-understandable.

1.4 eScience

The UK eScience program, launched by John Taylor (Director General of Research Councils), is designed to

> ...develop advances in scientific data curation and analysis and to be a primary source of top quality systems and repositories that enable management, sharing and best use of research data. [22]

eScience builds upon accessible structured knowledge resources and information, which is a severe limitation in eChemistry. This work examines methods to allow large volumes of chemical information to be converted rapidly to structured data by automated methods, requiring ideally no human intervention, thereby making it *machine-readable* and *machine-understandable*. One of the goals of the community is to create a Semantic Web of data, described in 2001 Tim Berners-Lee as:

> . . . an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation. [23]

The critical part of the above quotation is that *information is given well-defined meaning*. In relation to chemistry, this means that a computer must understand the concept of a solvent, for example, which requires the development of an ontology (a full description of the properties applicable to, and relationships between, each term). This is the difference between machine-readable and machine-understandable data. With these technologies in place, proponents of the Semantic Web assert that this will allow computers to reason [24, 25, 26].

The outline of a typical eScience experiment is shown in figure 1.8 in the most general terms possible; this form of experiment is commonly referred to as a workflow. It is important to note that it is not always necessary for all of the processes to be present for an experiment to take place — for instance if the data is already in an appropriate form then no translation process would be required.

eScience experiments often deal with large quantities of data (especially before the cleaning process) with the volume likely to increase in future, therefore, automation of the various processes (and methods of connecting them) is becoming increasingly necessary. This work examines all the components of such an experiment but in essence attempts to answer the question

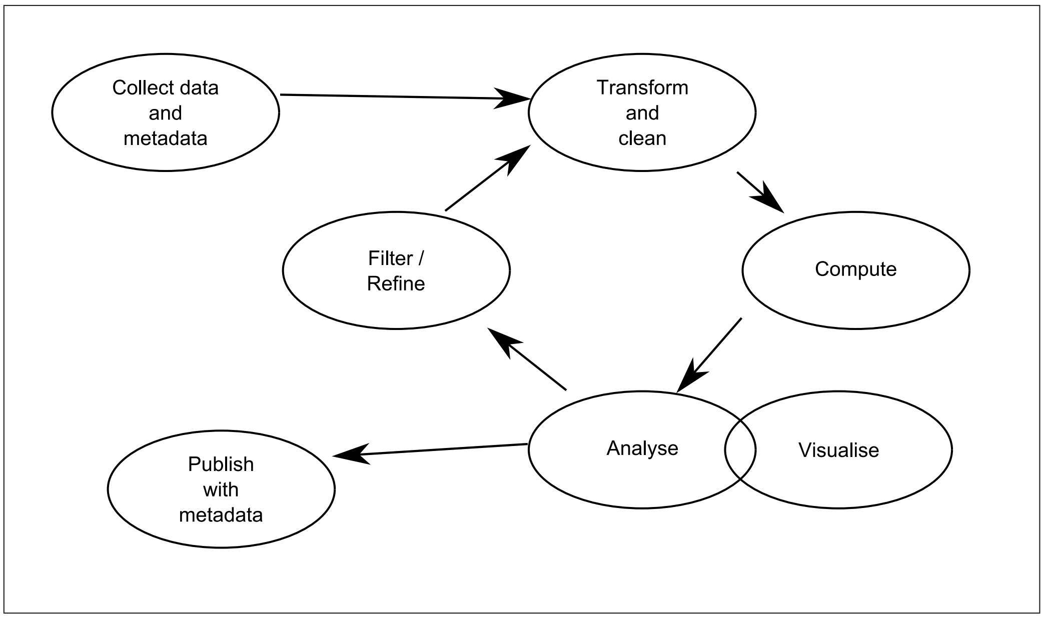


Figure 1.8: An outline of a typical eScience experiment — it is expected that many refinement cycles occur before the final publication.

> can a machine read the scientific literature as it is currently presented and re-use it for scientific research?

It should be noted that the final publication process does not necessarily indicate that an article in a journal will be the result. To illustrate a possible *experiment* imagine a chemist trying to answer the question;

> what percentage of the molecules with a reported synthesis published in the last year in Organic & Biomolecular Chemistry contained a non-conjugated ketone?

The collection of the data involves the automated reading of all the journal articles from the relevant year, the transformation and cleaning might require converting all the articles into a machine-understandable form and removing all those compounds that do not have a reported synthesis, the compute process would determine (perhaps by examining the molecular structure or infra-red spectrum) which of these contained a non-conjugated ketone, the analysis simply converts the counts of the molecules of each type to a percentage which is finally relayed to the chemist in a suitable manner.

Each process in the workflow can be thought of as a workflow in its own right, with a very similar structure to the overall experiment. In effect each process must consider

- how the data comes in
- how it can be translated to a usable form
- a transform of some sort is performed
- analysis of the transform occurs (for example, did the transform work)
- how should the data be emitted

1.5 Machine-Understandable Data

It is important to clarify what is meant by machine-understandable data and this can be done most easily by example. Figure 1.9 shows a picture of an organic reaction — this is human-readable and understandable (at least for someone acquainted with basic organic chemistry). Whilst it is machine-readable — a program has determined that the binary file holding the data should be rendered as a picture on screen — the data contained in the picture is not machine-understandable — the picture could be anything and would be treated the same way. To extract structured data from such a representation is difficult and methods for this are not considered in this work.

Figure 1.10 shows the same reaction in a text-based representation which is still human-readable, human-understandable and machine-readable. The data is still not machine-understandable in that the text could still be anything, but it is now in a more accessible form because text can be more easily manipulated and searched than the pictorial representation.

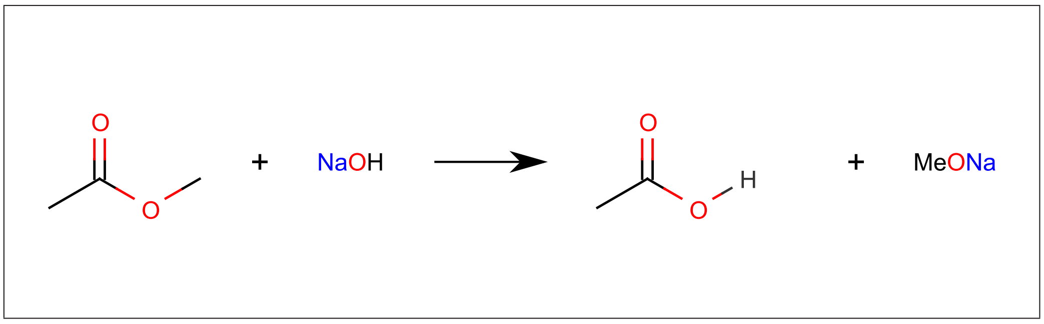


Figure 1.9: A chemical reaction represented pictorially.

Methyl ethanoate and sodium hydroxide react together to form ethanoic acid and sodium methoxide.

Figure 1.10: A chemical reaction represented in a text based form.

Figure 1.11 again shows the same reaction and again the representation is text-based but now contains XML *elements* that are used to *markup* the data. This representation is probably the least human-friendly. It contains a lot of information that the reader already knows which makes it difficult to read but it is still human-understandable. It is this (human-redundant) information which allows this representation to be machine-understandable, although it should be noted that simply marking up the data does not mean that the data is machine-understandable. However, if the syntax and semantics used to mark up the data are predefined then a chemically aware program can implicity understand that a reaction is occurring between the two reagents to form the two products.

Chemical Markup Langauge (CML) provides the set of elements and semantics required to hold most chemical data. Holding the data in this form also has the advantage that it is relatively easy to convert the data in this form to other forms — details of how this might be achieved and further background on XML and CML are given in section 1.10. The translation process shown in figure 1.8 therefore represents the conversion of data from

```
<reaction>
  <reagentList>
    <molecule>methyl ethanoate</molecule>
    <molecule>sodium hydroxide</molecule>
  </reagentList>
  <productList>
    <molecule>ethanoic acid</molecule>
    <molecule>sodium methoxide</molecule>
  </productList>
</reaction>
```

Figure 1.11: A chemical reaction represented using structured text.

another form into CML, a process commonly referred to as *parsing*. The other forms of data representation are considered below.

1.6 Formats for Reporting Chemistry

Chemistry is reported in a variety of formats ranging from highly structured documents with well-defined fields to narrative discourse. One of the goals of this study is to explore the feasibility of extraction of data into a semantically-rich and thereby machine-understandable form. In general, documents with more structure are simpler to parse because there are fewer ambiguities and many of the underlying semantics are given explicitly. The formats covered in this thesis fall into two main categories: human-authored data and data relating to computational chemistry programs which is usually machine-authored. An example of human-authored data has already been encountered in figure 1.2. The techniques developed to parse data in this form are examined in chapter 2.

The data relating to computational chemistry programs can be split into a further two categories: program input and program output. Input data (figure 1.12) is inherently machine-readable and must be machine-understandable, albeit only by the specific program. Techniques adapted from compiler theory allow such data to be transformed into more generally machine-

```
[ atoms ]
; nr type resnr resid atom cgnr charge
1 O 1 DRG O8 1 0.000
2 CB 1 DRG C3 1 0.000
                                          ⋮
8 H 1 DRG HAB 1 0.280
[ bonds ]
;ai aj fu c0 c1
1 2 1 0.123 502080.0 0.123 502080.0 ; O8 C3
                                          ⋮
7 8 1 0.100 374468.0 0.100 374468.0 ; N4 HAB
```

Figure 1.12: A section of a GROMACS topology file, representing data in a machine-readable and machine-understandable form.

understandable forms (see chapter 3). Program output (figure 1.13) is generated by a machine and thus consists of a finite vocabulary. Typically this has well-defined structure but error messages may appear at any point making perfect parsing almost impossible. It is designed to be human-readable and human-understandable but not necessarily machine-understandable. Parsing data in this form required the development of new methods, the details of which are given in chapter 4.

1.7 Compute Processes

There are two compute processes involved in this work reflecting the two major sources of data considered. The first process involves checking that the analytical data reported for a molecule is self-consistent and reasonable, the second uses computational chemistry programs to find an optimised structure of a molecule and requires a little more introduction.

Quantum chemistry has emerged as an important tool for investigating a wide range of problems in chemistry. With the development of computational

```
*******************************************************************************
**                            MOPAC2002 (c) Fujitsu                          **
*******************************************************************************
*                  MOPAC2002 Version 1.01     CALC.'D. Mon May 19 01:25:16 2001
*  PM5      - THE PM5 HAMILTONIAN TO BE USED
      46
ATOM    CHEMICAL     BOND LENGTH    BOND ANGLE    TWIST ANGLE
NUMBER    SYMBOL      (ANGSTROMS)    (DEGREES)     (DEGREES)
   1       Cl           0.000000       0.000000       0.000000
   2        N           5.069306  *    0.000000       0.000000        1
    46
  48        H           1.090015  * 109.466551  *   54.661165  *    23    22    18
          EMPIRICAL FORMULA: C18 H24 N3 Cl3
     MOLECULAR POINT GROUP   :   C1
        SIGMA BONDS               49
        LONE PAIRS                12
         FINAL HEAT OF FORMATION =         -0.90975 KCAL =         -3.80639 KJ
         ELECTRONIC ENERGY       =     -30264.46015 EV  POINT GROUP:     C1
         CORE-CORE REPULSION     =      26026.82711 EV
         MOLECULAR WEIGHT        =    388.767
```

Figure 1.13: A section of a MOPAC output file.

methods and the availability of more powerful computers, it has become possible to solve chemical problems that until relatively recently were impossible. Quantum-mechanical methods are now routinely applied to problems related to molecular structure and reactivity [27].

There are three common approaches to computational chemistry: *ab initio* methods, semi-empirical methods and density functional theory (DFT). As described in section A.1, *ab initio* methods are 100% mathematical, meaning that all of the information generated about an atom, molecule or reaction comes from the fundamental quantum mechanical calculations (specifically, the Schrödinger equation). This requires significant computing resources, hence most calculations are limited to small molecules, typically those consisting of less than 100 atoms. Semi-empirical methods provide a way to study larger molecules. As the name suggests, semiempirical methods are a combination of *ab initio* methods coupled with the use of data from empirical studies — they are based on the Hartree-Fock formalism but make many approximations and obtain some parameters from empirical data rather than from theoretical principles. DFT is often considered to be an *ab initio* method for determining molecular electronic structure, despite the fact that most of the common functionals use parameters derived from empirical data.

It is usual for the limiting factor in calculations to be the amount of time available for the calculation to run. It is therefore important to be able to predict how long a particular calculation will take. The ability to do this becomes increasingly important if the process is to be automated. The background and mathematical forms (where appropriate) of the three methods are given in appendix A. From equation A.25 it is seen that basic *ab initio* methods require a four-way integration over all the basis functions to be performed; such methods are expected to scale as n^4 where n is the number of basis functions. Improvements to the basic theory such as second order Møller-Plesset perturbation theory (MP2) or coupled cluster single and double excitation calculations (CCSD) scale to higher powers (n^5 and n^7 respectively). DFT implementing a standard coulomb integral scales as n^4, but only as n^3 when using an auxiliary basis. However, in the limit of a large molecule, the pairwise interaction dominates (which scales quadratically) and the overall behaviour typically scales as $n^{2.2}$. Semi-empirical methods tend to scale as n^2 [28]. The use of these scaling relationships in predicting calculation times is shown in sections 5.5 and 8.6.

1.8 Analysis and Visualisation

The process of analysing the data in an eScience experiment can vary greatly in complexity and in some cases can be almost trivial. For instance, the analysis process in chapter 2 only requires the list of problems found in the analytical data to be sorted in order of decreasing severity. It is often the case that the type of analysis required will depend largely on the visualisation and the manner of publication. There have been two approaches taken in this work and these are discussed separately below.

The Experimental Data Checker (EDC) and OSCAR programs were originally designed to aid chemists and editors finding mistakes in the analytical data; the intention was to create a system where possible errors in the data were highlighted for subsequent human curation. The publication process therefore involves the reporting of the possible errors to the user. The visualisation process began as a plain text list of the errors found, although subsequently, to aid comprehension, these errors were highlighted in different colours (reflecting the severity of the perceived error). As the program evolved, a tool to recreate the various reported spectra was incorporated into the visualisation process and the user was given the ability to view the analysed data in various forms. All the various parts of the programs were written in Java (see section 1.13) and further details are given in chapter 2.

The analysis process in the remaining chapters focuses mainly on comparing the three dimensional structures of molecules before and after a geometry optimisation calculation, thereby allowing the refinement of the cleaning process. The geometries are compared primarily by examining the change in bond length, angle and torsion between the two structures, usually by using graphs to detect anomalies and trends. Unfortunately no tools were found that were able to represent the data in the required manner, hence a graphing tool was written to allow the appropriate visualisation (see section 1.12.1). This program allowed each data point to be linked to an external web page which contained the input and output structures in Jmol applets (see section 1.14). Jmol is a molecular viewer which allows the user to manipulate the structure in three dimensions (by changing the orientation for example) and was used to visually compare the molecules. The manipulation of the data and creation of the web pages was again performed by programs written in Java and designed to be generally applicable.

1.9 Publication

Clearly the ultimate publication of this work is as a thesis, although, owing to the constraints of the paper medium, the work will finally be published in a more interactive format. However, there are multiple publication processes utilised in chapter 3 onwards, for instance; the inability to parse program output using only compiler theory techniques, is a result. Similarly, the archival of the machine-understandable form of a document (automatically allowing the re-use of the data) can also be interpreted as part of the publication process. These results are seen as byproducts, albeit useful byproducts, of the final publication which is a set of tools and protocols that allow high quality data to be extracted and reused for research.

An introduction to the underlying tools and technologies that are used throughout this work for holding, manipulating, visualising and analysing the data is given below.

1.10 eXtensible Markup Language

eXtensible Markup Language (XML) [29] is based on Standard Generalised Markup Language (SGML) [30], the international standard for defining the descriptions of the structure and content of different types of electronic content (standardised in ISO-8879:1986). SGML was created by Goldfarb in the 1970s and is a very powerful metalanguage whose primary purpose is to create other markup languages, such as HyperText Markup Language (HTML) [31]. A markup language defined using SGML or XML has a specified vocabulary (the labels for elements and attributes) and a declared syntax (the grammar defining the hierarchy and other features). XML first appeared in 1996 with the first World Wide Web Consortium (W3C) recommendation published in 1998. The W3C is a vendor-neutral body which

> specifies protocols for the Web infrastructure and develops interoperable technologies (specifications, guidelines, software and tools) to lead the Web to its full potential. [32]

The major advantage of XML over HTML is the ability to define whatever *elements* are necessary to express and support the requirements of the application (see figure 1.14) as opposed to the elements[¶] in HTML which are

[¶]The *tag* is the text between the angle brackets, *i.e.* `<tag>`, whereas the *element* is the start and end tag and all the content between them. The terms are often used interchangeably and the meaning should be apparent from the context.

```
<molecule id="HCl">
  <atomArray>
    <atom id="h1">
      <string builtin="elementType">H</string>
    </atom>
    <atom id="Cl1">
      <string builtin="elementType">Cl</string>
    </atom>
  </atomArray>
  <bondArray>
    <bond atomRefs="h1 Cl1">
      <string builtin="order">1</string>
    </bond>
  </bondArray>
</molecule>
```

Figure 1.14: An example of an XML document.

predefined.

1.10.1 Validation

All XML documents must be well-formed[‖]; they may also be valid. Validation is performed against a Document Type Definition (DTD) or, more recently, a XML Schema [34, 35].

A DTD defines which elements are permitted for documents of that type, what their names are, where they may occur, if they are optional or required, what types of values they can hold (although this mostly applies to *attributes* not elements) and how they are related to each other (for example; parent, child or sibling). XML Schemas also enable specifications to be placed on the type of data present in any given element or attribute.

1.10.2 Namespaces

The ability for every author and user of XML to create their own element names means that tag names may be replicated but the meaning might be very different in each case. Such ambiguities may not be resolved even if

[‖]Some browsers allow HTML documents not to be well-formed although this can lead to ambiguous interpretations; eXtensible HyperText Markup Language (XHTML) must be well-formed [33].

both tags are defined in a DTD. To prevent any collision of DTDs the W3C has defined a mechanism for *namespacing* each document definition [36]. This process allows each XML name to be globally unique; this is achieved by mapping a namespace attribute to a Uniform Resource Identifier (URI). These URIs exist solely to provide a globally unique string and need not represent a physical Web address, and furthermore do not require an Internet connection to function. It is common for URIs to be based on the creator's domain name (to provide the necessary uniqueness). This also allows separate vocabularies and ontology to be defined for each area (for example, CML or Math Markup Language).

Currently only elements may be namespaced but the use of dictionaries or ontologies — which represent formal descriptions of concepts — has required the ability to namespace individual attributes. The Scientific Technical Medical Markup Language (STMML) provides the basic concepts for creating dictionaries. STMML was conceived by Murray-Rust and Rzepa and is a domain-independent language designed to manage the infrastructure of (mainly numeric) disciplines [37].

There are no limits to the number of dictionaries that can be associated with a given application and references to the dictionaries are identified by a prefix (*e.g. iucr:*). The use of dictionaries can greatly improve human comprehension of the data in a document (once converted to a human readable form). This might be realised by showing the reader the corresponding dictionary entry for a particular data item on *mouse over*[**].

Data definitions are sometimes referred to as *metadata*, thus the fact that an XML document conforms to a specified DTD or Schema might be an example of metadata. The ability for XML to store metadata (such as how to handle the corresponding data) allows machines to understand the data and also makes it inherently easier to retrieve a particular data item. The Dublin

[**]Mouse over means that an action is initiated when the mouse pointer is held over a defined area of the screen.

Core Metadata Initiative is an organization engaged in the development of interoperable online metadata standards [16]. The recommendations of this initiative have been formally endorsed [38, 39] and include the ability to specify metadata such as the creator of the data, contributors, associated dates, rights held in and over the data and the location of related data. There are also numerous entities for describing how the data was collected and archived.

1.10.3 Data Display

Unlike HTML, in which almost all the elements affect only the presentation of the document and hence make it more human-readable, XML has been primarily designed to hold and pass information in a machine-understandable form. To allow data held in XML to be easily read by humans it is usual to apply a stylesheet to the document. A stylesheet essentially tells the computer how each tag, or group of tags, should be processed; multiple stylesheets can exist for any given document [40].

eXtensible Style Language (XSL) is used to define a set of primitives which describe a document transformation, conversions using this method usually refer to XSL transformations (XSLT). XSL provides a versatile and powerful language for transforming an XML document into something else, using complex transformations and dynamic operations. XSLT are based on pattern matching: each rule specifies a particular action that is performed when the associated pattern is encountered in the source document. XSLT can be used to merge documents, applying various filters to documents, inter-convert between various XML *dialects*, and sort a document on its content. Figure 1.15 shows the workflow for a XSLT.

1.11 Chemical Markup Language

Murray-Rust and Rzepa first presented the Chemical Markup Language (CML) to the world at the 1995 American Chemical Society August Meeting in Chicago [41, 42]. The following year the W3C began work on the XML

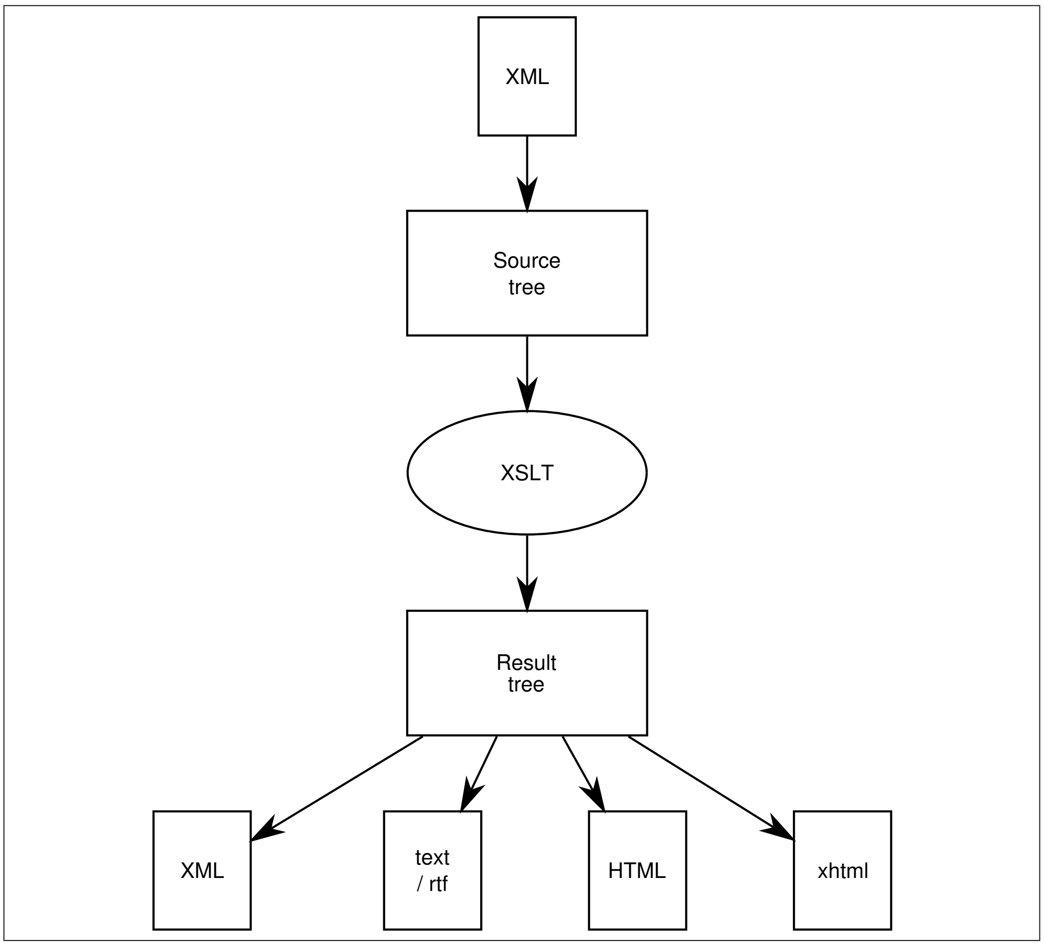


Figure 1.15: XSLT flow; the XML is read and a corresponding tree structure created, the XSL transforms are then applied to the tree structure to create the result tree which is subsequently translated into the required output format.

project and in 1997 CML became the first ever XML DTD. The first version of the CML (CML 1.0) specification was formally published in 1999 [43]. CML is an extensible base for chemically-aware markup languages.

Historically, CML only focused on molecules (*i.e.* discrete entities representable by a formula and, usually, a connection table). It supported a hierarchy for molecules as well as reactions and macromolecular structures or sequences. It allowed for quantities and properties to be specifically attached to molecules, atoms or bonds, but originally it had no support for physicochemical concepts, reaction schemes, mechanisms, reactive centres, or spectator molecules.

CML was developed to support both the presentational aspects and semantic content of chemistry. It is designed to be ontologically neutral. This neutrality is important as most molecular file formats, such as the MDL molfile [44], contain complex, often implicit, ontologies that are not necessarily convertible into other formats. By keeping CML ontologically neutral, this problem is minimised and as such, it is possible to convert CML into other formats. CML also uses abstract data types wherever possible. It can be seen that a melting point has the same abstract structure as the price of an item or a person's age. This data type might be described as a floating point number, with units, allowed range and links to metadata. This approach greatly widened the applicability, support and tools available for CML.

CML was designed to be fully compatible with XML and to re-use its ideas and technologies. As such it captures the content of chemistry rather than the presentation. CML was originally cast as a DTD, but with the advancing technologies of XML and XML Schema Language it was decided that CML needed to be recast in a more tightly modular system. This led to the creation of CML 2.0 [45]. CML 2.0 also aimed to address additional aspects that CML 1.0 ignored, *e.g.* chemical substances, and also to describe some of the elements in more detail, such as formula and electron, which were not well-described in CML 1.0.

A modular approach has been taken for the language specifications. This allows subsets of the language to be used independently — for instance CMLComp represents concepts of computational chemistry. The use of dictionaries allows CML to be extended still further. The following CML modules are available:

STMML domain-independent specification for general scientific data (including units, metadata, dictionaries, data types and data structures)

CMLAll the complete CML language definition.

CMLCore The core part of molecular structure representation.

CMLComp CML for computational chemistry

CMLReact Support for chemical reactions including enzymes

CMLSpect CML for spectra including NMR, MS and infra-red; this interoperates, rather than competes, with the more formal industry activities such as AnIML [46], GAML [47] and SpectroML [48]

CMLSnap CML to describe and handle dynamic (animated) reactions

CMLQuery A query language for chemistry which is currently under development

CMLCM condensed matter systems, also under development

CML is generally considered to be the *target* format for data storage in this work. In other words we try to translate data into a form that is represented using the elements, attributes and other data items as specified in CML. In some cases this is done via the application of stylesheets to intermediate XML representations.

1.11.1 JUMBO

CML does not provide chemical perception; it is only designed to hold data. JUMBO began as the Java Universal Molecular Browser for Objects but subsequently evolved into a generic name for the software that manages schemas for CML [49]. The JUMBO package can also be used to check the chemical validity of the data held in a CML document or data being added to an existing document. For instance, if a user were to attempt to specify a new bond in a molecule, JUMBO will check that all the atoms involved in the bond exist, are part of the molecule and (if desired) whether they are within reasonable bonding distance.

JUMBO initially implemented the CMLDOM (DOM is a Document Object Model [50, 51]) to provide the required restrictions on data and data types that were unavailable using the CML DTD [52]. Since the introduction of XML Schemas this is no longer as necessary — the base classes for JUMBO are now created directly from the CML Schema. JUMBO is structured such that the data structure is separated from the tools as proposed by Knuth [53]. The program is Open Source [54] and available on the SourceForge repository [55]; this work has used both JUMBO 4.6 and JUMBO 5.3.

1.12 Scalable Vector Graphics

Scalable Vector Graphics (SVG) is a language for describing two-dimensional graphics and graphical applications in XML [56]. SVG 1.1 became a W3C Recommendation in January 2003 and forms the core of the current SVG developments. Sun Microsystems [57], Adobe [58], Apple [59], IBM [60], and Kodak [61] are some of the organizations that have been involved in defining SVG.

Advantages of using SVG over other image formats (such as Joint Photographic Experts Group (JPEG) and Graphic Interchange Format (GIF)) are:

- SVG files can be read and modified by a large range of tools, including any text editor
- SVG files are smaller and more compressible than JPEG and GIF images
- SVG images are scalable
- SVG images can be printed with high quality at any resolution
- SVG images are *zoomable* — any part of the image can be magnified without degradation
- text in SVG is selectable and searchable
- SVG works with Java technology
- SVG is an open standard
- SVG files are pure XML
- all the attributes of SVG elements can be animated

The main competitor to SVG is Adobe Flash. The biggest advantage SVG has over this is the compliance with other standards (*e.g.* XSL and the DOM) whilst Adobe Flash relies on proprietary technology that is not Open Source.

SVG is an application of XML and as such is compatible with the XML1.0 recommendation; it uses the XML Linking Language (XLink) [62] for URI referencing and requires support for base URI specifications as defined in XML Base [63]. The content of an SVG document can be styled using either CSS or XSL, where external stylesheets can be referenced [64]. SVG includes a complete DOM (level 1) and supports or incorporates many of the facilities described in the DOM level 2 specification including the CSS object model and event handling. The animation features incorporate and extend the general-purpose XML animation capabilities described in the SMIL Animation specification [65, 66].

1.12.1 SVG Graphing

Scientific analysis often involves the creation of graphs to detect trends or outliers. Whilst graphs can be easily and quickly created using applications such as R [67] or Microsoft Excel [68], the resultant graphs are neither interactive nor can they be immediately mounted as web pages. To simplify the analysis process, a program was written to create graphs in SVG. This application can generate basic x-y plots, histograms, smoothed density plots and quantile-quantile (QQ) plots.

The primary motivation for the creation of this software was to allow any data point (or anything on the graph) to be linked to an external document — in this work this has usually been a web page displaying the 3D representations of the molecules — which displayed further information about that point. The user is also permitted to specify associated data for a point, which can be anything that is representable by a string. This data is displayed when the mouse pointer is over the area of the graph represented by that point.

The use of SVG to create graphs allows the immediate integration into web pages — whether or not the interaction functionality is used — as well as ensuring that there is no loss of quality if a user desires to examine a particular area of the graph at a larger magnification. The text in any SVG document is searchable and therefore all the data and metadata contained in a graph mounted in or as a web page can be indexed by internet search engines.

The SVG graphing application was designed to be used either from within a Java programming environment or as a *standalone* program with a graphical user interface (GUI). The options available for the presentation of the graph are based on those provided by R and most are accessible through the GUI. The application is currently in a functional state and was used to analyse all the data produced during this work but remains a work in progress, requiring greater CML support (especially CMLTable) and synchronising with the pelote specification [69].

1.13 Java

Java [70, 71] consists of three, equally important parts: the Java language, the Java Virtual Machine (JVM) and the Java platform. Java is a 'write once, run anywhere' technology, *i.e.* so long as the destination system has a JVM, the program will run on that system. This makes Java a very versatile and portable language.

The Java programming language is object-oriented — object-oriented programming (OOP) may be seen as a collection of cooperating objects, as opposed to a traditional view in which a program may be seen as a list of instructions to the computer. In OOP, each object is capable of receiving messages, processing data, and sending messages to other objects. Each object can be viewed as an independent little machine with a distinct role or responsibility.

OOP is intended to promote greater flexibility and maintainability in programming, and is widely popular in large-scale software engineering. By virtue of its strong emphasis on modularity, object-oriented code is intended to be simpler to develop and easier to understand subsequently, lending itself to more direct analysis, coding, and understanding of complex situations and procedures than less modular programming methods.

The developers of Java tried to make the language powerful, but also to avoid overly complex features that can bog down an object-oriented language. By keeping the language simple, it is easier to write robust and (hopefully) bug-free code. The JVM is also known as the Java interpreter. Without the virtual machine the code will not run on a system, as it is this virtual machine that interprets and runs the code. The Java platform is important in that all Java code relies on the set of predefined classes (modules of Java code that define a data structure and a set of methods that operate on that data) that comprise the Java platform.

Java classes are organised into packages (related groups) and the Java platform defines packages for functionalities such as input/output, networking, graphics and regular expressions. The most common Java programs written are applets and applications. Applets are programs that adhere to certain conventions, allowing the program to run within a Java-enabled browser. An application is a standalone program that runs directly on the Java platform. Java is also designed to be a powerful software platform.

Wherever possible, all the tools created as part of this work have been written in Java and in such a way that they should be reusable in other applications. There are already several instances of classes and methods being incorporated into, or used by, programs written for other purposes. The programs are entirely Open Source and are available to all [54].

1.14 Jmol

Jmol is a free, Open Source, molecule viewer written in Java and available as both an applet and an application [72]. Jmol is capable of extracting the 3D coordinates of molecules stored in various file formats including CML, CIF and GAMESS. It also allows the user to export the rendered molecule to graphical formats including PovRay [73]. All the 3D images of molecules in this work have been rendered using Jmol to create the PovRay file which was subsequently stored in the encapsulated postscript format. Jmol applets were also used throughout the analysis of the data to visually compare the geometries of molecules.

Chapter 2

The Quality of Data in the Chemical Literature

Languages are used to communicate information; chemistry is a language without native speakers and has developed as a written rather than spoken one. This leads to a level of communication that is adequate but not optimal. The work presented in this chapter examines techniques to extract data from the literature in its current form into more comprehensible formats for both humans and machines and to validate and re-use.

The most common method to disseminate results in the chemistry field is via publication in peer-reviewed journals. This process is designed to allow only valid and accurate science and data into the public domain. Some errors (albeit mostly minor) can, and do, make it through the process and into the published work. These errors are often trivial typographic mistakes, but others are more serious and may lead to the work being withdrawn from publication. In future, the increase in volume of articles taken in conjunction with human error must result in an increased number of mistakes (and possibly more serious ones than at present) permeating through the process.

Typographic errors are generally trivial for a human reader to absorb and correct without actually being aware of them. For example the recent meme that was found on many blogs and in inboxes

Aoccdrnig to rscheearch at Cmabrigde Uinervtisy, it deos not

> mttaer in waht oredr the ltteers in a wrod are, the olny iprmoatnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a total mses and you can sitll raed it wouthit a porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe.

should actually read

> According to research at Cambridge University, it does not matter in what order the letters in a word are, the only important thing is that the first and last letter be at the right place. The rest can be a total mess and you can still read it without a problem. This is because the human mind does not read every letter by itself, but the word as a whole.

Whilst a human can read and interpret the quotation, a computer will not in general be able to do so. The original quotation contains 34 unique spelling mistakes (*wrod* and *taht* are both repeated twice). The quotation was run through the spell checking tools from two different authoring programs (WinEdt [74] and Microsoft Word [75]). A list of alternatives to the misspelled word are suggested by both programs. The words at the top of the list are considered by the program to be more likely alternatives, consequently, the results have been broken down into three categories;

Top The correct word is at the top of the suggested alternatives;

Present The correct word is present in the suggested alternatives, but is not the most likely;

Absent The correct word does not appear in the suggested alternatives.

The results are presented in table 2.1; whilst both programs correctly identified the intended word in over 75% of the cases neither program would have been able to reconstruct the entire sentence correctly.

	Microsoft Word	WinEdt
Top	27	26
Present	1	4
Absent	6	4

Table 2.1: Comparison of spell checking between Microsoft Word and WinEdt.

The quotation, as originally cited, is not machine-understandable — that is, it would be impossible for even a natural language processing program to determine the meaning of the quotation. The increased desire for automation of the categorisation, searching and interpretation of the published literature requires that the literature be machine-understandable. Although it is extremely unlikely that any published literature would contain the kind of gross errors present in the quotation above, a minor error (for example in an abstruse chemical name) is far more likely.

2.1 Information Extraction

In computer science, information extraction (IE) or text mining is a type of *information retrieval* where the goal is to automatically extract structured information from unstructured machine-readable documents; an example of IE can be found in the analysis of gene expression data. An association rule represents a set of items that are likely to be seen together; for example, the rule

$$\{\text{cancer}\} \Rightarrow \{\text{gene A}\uparrow, \text{gene B}\downarrow, \text{gene C}\uparrow\}$$

states that whenever cancer was found, gene A and gene C were highly expressed, but gene B was highly repressed and all three genes occurred together [76]. Typical subtasks of IE are:

Named Entity Recognition recognition of entity names (for people and organizations), place names, temporal expressions, and certain types of numerical expressions;

Coreference identification chains of noun phrases that refer to the same object. For example, anaphora is a type of coreference;

Terminology extraction finding the relevant terms for a given corpus.

The development of ontologies, and hence text mining, in other spheres has advanced rapidly, particularly in the biosciences. Bioinformatics has embraced the concept of literature data mining and much work has been done in this area ranging from the simple recognition of terms to the extraction of interaction relationships from complex sentences [77, 78]. This thesis details how such methods can be applied to chemical documents to create structured, semantically-rich, machine-understandable versions of the documents as well as describing the development of various text mining processes and chemical ontologies.

The extraction of data from legacy formats is simplified by increasing structure, thus if every chemist were to write articles in XML (with a predefined DTD or Schema) all the concepts would be immediately recoverable. For example an infra-red spectrum of butan-2-ol might be represented as;

```
<spectrum title='butan-2-ol' type='infrared' state='gas' >

                              ⋮

</spectrum>
```

However, many synthetic organic chemists use Microsoft Word or a similar program as their authoring tool. These programs tend to focus on the presentation of the data, rather than content of the data. Whilst it might be unreasonable to expect authors to create their documents in XML with fully integrated CML, the advent of online publication makes a more integrated version of an article (a datument [79]) more attractive [80]). Even if datuments become the accepted publication method there will still be a huge legacy corpus which will require extraction.

2.2 The Experimental Data Checker

The Royal Society of Chemistry (RSC), recognising this potential problem, sponsored a project* to develop a tool both to improve the quality of manuscripts submitted to them for publication and to allow reviewers to detect errors more easily. Initially, two students (F. R. Norton and this author) were involved for a three month period to develop a proof-of-concept Experimental Data Checker (EDC).

The brief for the EDC project was to analyse a corpus of typical organic chemistry articles — both those submitted to the RSC for publication and those already published — supplied by the RSC, and to determine to what extent the extraction of data was a tractable problem. A proof-of-concept cross-platform program that could extract (to a greater or lesser extent) the data present was to be developed which would enable human reviewers to detect errors more easily. It was envisaged that this would involve performing self-validation tests on the data in the analytical data section. The RSC also requested that a DTD be created to represent the extracted data. The specifications for the project required that the checking process should not affect the way in which the author could create the manuscript, and, that only a minimal amount of extra work would be involved in the checking process†.

2.2.1 The legacy format of organic chemistry articles

An analysis of the corpus provided showed that most organic chemistry articles (and articles for submission in the journal) are divided into well-defined sections. The published article structure is shown in figure 2.1. The same sections are found in the unpublished articles but often in a different order. Examples of each section are given below.

*The project was collaborative; S. E. Adams, F. R. Norton, C. A. Waudby and this author have all been involved; overall this author contributed approximately 50% of the work. Where work was performed by a specific individual this is indicated.

†Such a design brief — that the user should not be expected to do significant further work — is sometimes referred to as the requirement for a Big Red Button solution.

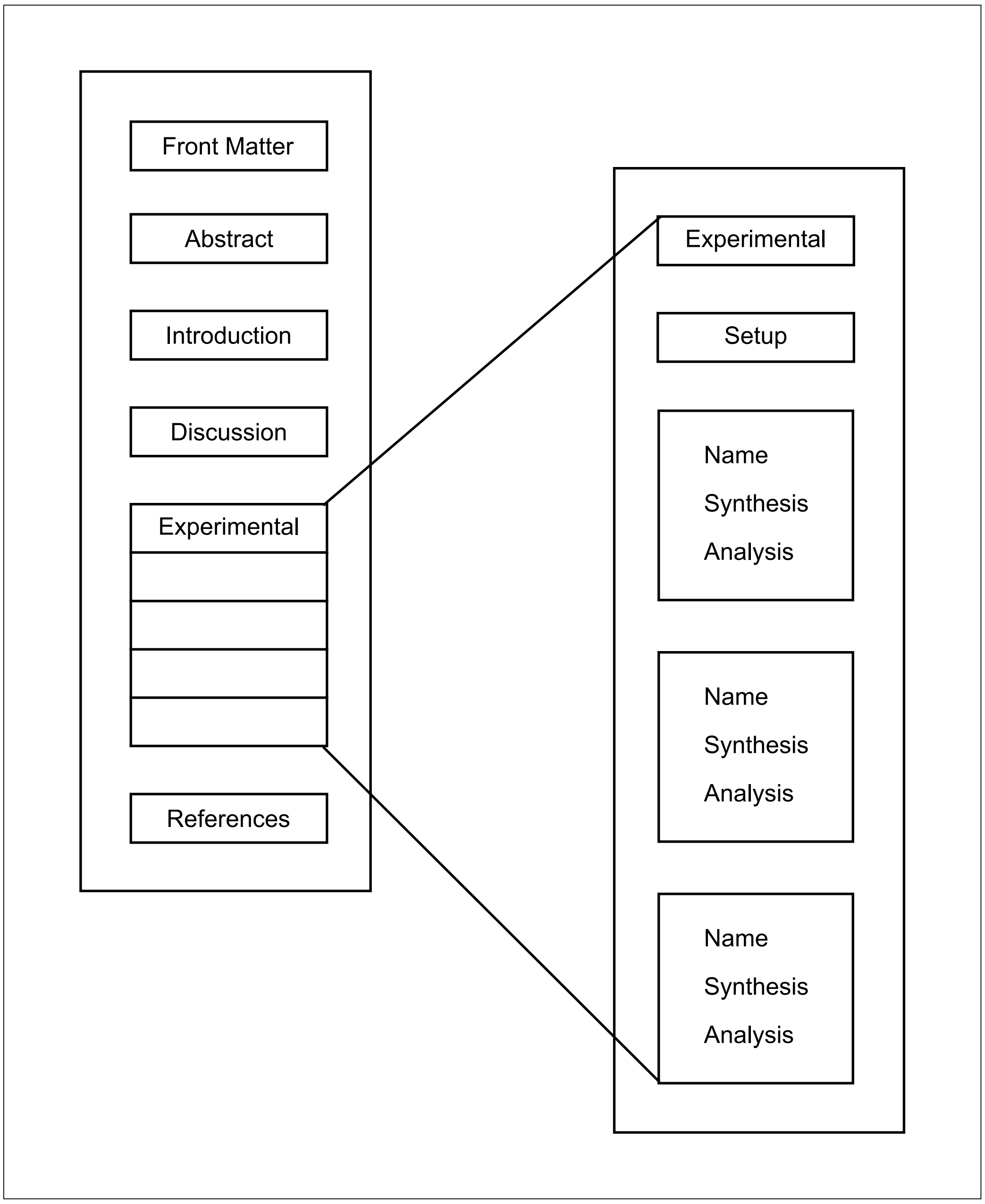


Figure 2.1: The typical structure of an organic chemistry article.

. . . Mp 62–65 °C. FTIR (NaCl, thin film) 3025 (m), 2952 (s), 2881 (m), 1734 (s), 1707 (s), 1249 (m), 1184 (m), 1132 (m), 1054 (m), 950 (w) cm^{-1}. ^1H NMR (100 MHz, $CDCl_3$) δ 1.10 (m, 1H), 1.25 (m, 6H), 1.33 (s, 3H), 1.61 (m, 6H), 2.07 (dt, J = 13.5, 5.1 Hz, 1H), 2.41 (m, 1H), 2.59, (dt, J = 14.3, 6.2 Hz, 1H), 2.77 (dd, J = 12.0, 2.8 Hz, 1H), 3.92 (m, 4H), 4.19 (m, 2H). ^{13}C NMR (400 MHz, $CDCl_3$) δ 210.6, 173.2, 112.7, 65.7, 65.3, 61.5, 61.5, 45.3, 42.3, 35.0, 30.8, 29.4, 23.6, 23.0, 17.4, 16.3, 14.6. $[\alpha]_D^{20}$ +11.1 (c, 7.15, $CHCl_3$). Anal. Calcd for $C_{17}H_{26}O_5$: C, 65.78; H, 8.44. Found: C, 66.07; H, 8.35.

Figure 2.2: An example of typical analytical data found in an organic chemistry paper [82].

5',6',8a'-Trimethyloctahydro-2'H-spiro[[1,3]dioxolane-2,1'- naphthalene]-5'-carbaldehyde (**20**). A CH_2Cl_2 (10 mL) solution of alcohol **19** (270 mg, 1.01 mmol), 4-methylmorpholine N-oxide (130 mg, 1.10 mmol), and 4 Å MS (300 mg) was stirred for 10 min. At this time, tetra-n-propylammonium pyruthenate (17.5 mg, 0.05 mmol) was added in one portion and the reaction was stirred for 1 h. The reaction was found complete by TLC, passed through a pad of silica (1 × 20 cm^2 with 1:1 hexanes:Et_2O), and concentrated to provide aldehyde **20** (260 mg, 98% yield) as a clear oil.

Figure 2.3: An example of a typical synthetic methodology found in an organic chemistry paper [82].

Analytical Data

Analytical data (figure 2.2) are physical properties used to determine the identity of a compound, such as NMR spectra and elemental analysis. When reported in journals or theses this data is highly formalised. The format is presentational rather than semantic and may include human-created errors arising from transcription, omission, spelling mistakes [81], unforeseen microstructure and vocabulary. Ambiguity is often present; hence recall may not be perfect. This is an attractive area for consideration because the data is semi-structured and when extracted would allow error checking (for self-consistency).

Starting from commercially available 3-furaldehyde, we accessed the known homoallylic alcohol 11 in 82% yield and 93% ee through Brown's asymmetric allylboration (Scheme 2). Alternatively, alcohol **11** could be prepared in 82% ee and 64% yield in a proline oxide-catalyzed asymmetric addition of trichloroallyl silane into 3-furaldehyde. In either case, O-alkylation of **11** with 2,3-dibromopropene **12** afforded allylic ether **13** in 75% yield. Lithium-halogen exchange followed by isopropoxy borolane **14** trap provided the pinacol borolane **10**, which was carried on to the borolane fragment **9** through a RCM by using Grubbs second generation catalyst **15** in 45% overall yield for the two steps. With access to pyran **9**, attention was turned toward generating coupling partner **6**.

Figure 2.4: An example of a typical chemical discourse found in an organic chemistry paper [82].

Synthetic Methodology

The synthetic methodology (figure 2.3) comprises a highly formal language and many stock phrases that may be valuable for tokenising. This format requires a *lexicon* to allow entity and microstructure recognition; shallow parsing and part-of-speech recognition techniques must also be employed. However, currently CML does not have the required elements or semantics to describe such data.

This data represents an extremely large resource which, if it can be searched more effectively (for instance on reaction type), can be made vastly more useful for both eScientists and the more traditional chemist. If the semantics of this area can be well-defined it should be possible for robots to entirely reconstruct (and re-perform) the synthesis. Parsing the synthetic methodology did not form part of the original EDC project but subsequent work has shown this area to be somewhat tractable.

Chemical Discourse

Chemical discourse encompasses, for example, the results and discussion section of an article (figure 2.4). This requires deep parsing and a substantial

The marine sponge metabolite (+)-cacospongionolide B (+)-1 is a member of a class of compounds bearing a γ-hydroxybutenolide moiety. This functionality has been suggested to be important in the inhibition of several forms of secretary phospholipase A_2 (sPLA$_2$), enzymes involved in events leading to inflammation. Given the role of chronic inflammation in diseases such as asthma, psoriasis, cancer, atherosclerosis, and rheumatoid arthritis, it is becoming increasingly important to discover and develop more effective agents that can mediate these pro-inflammatory signaling events. With a successful route to cacospongionolide B already in hand, efforts have turned toward understanding the structural features of the natural product responsible for inhibiting sPLA$_2$ activity. Our previous findings indicated that furan **2** possessed comparable sPLA$_2$ inhibitory activity to (+)-**1**, while the enantiomer of the natural product (−)-**1** was less active. In addition, several unnatural diastereomers of the natural product were identified that displayed improved sPLA$_2$ inhibitory activity over (+)-**1**.

Figure 2.5: An example of a typical narrative discourse found in an organic chemistry paper [82].

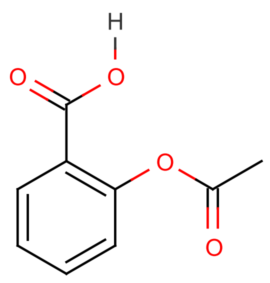
English lexicon as well as a chemical one. There are many reserved words that may allow parsing but currently this is outside the scope of this work.

Narrative Discourse

Narrative discourse comprises, for example, the introduction section of an article (figure 2.5). This also requires deep parsing and a complete English lexicon as well as a chemical one. This is outside the scope of this work.

Chemical Names

Chemical names are often expressed in a formal language and as such should follow specifications. However these often contain ambiguity and in some cases allow multiple (and equally valid) names to be given to a species (see figure 2.6). Systematic chemical names are not designed to be machine-understandable but may be described by a grammar. Non-deterministic methods must be used to parse chemical names in general, allowing am-



IUPAC name	2-Acetoxy-benzoic acid
CAS Index name	Benzoic acid, 2-(acetyloxy)-
Trivial name	Salicylic acid
Trivial name	Aspirin

Figure 2.6: Various names for the same connection table.

biguities to remain unresolved for as long as possible. The ability to parse a systematic chemical name is essential as it is often the only way in which a connection table for the compound can be recovered from a paper. The identification and parsing of chemical names is explored in section 2.4.

2.2.2 Common Chemical Concepts

The analysis section of each compound were examined and the most common well-defined concepts identified. These concepts were found to be:

- chemical name
- yield
- boiling point / melting point
- proton nuclear magnetic resonance (HNMR)
- carbon nuclear magnetic resonance (CNMR)
- infra-red spectrometry

- (high resolution) mass spectrometry (HRMS)
- elemental analysis
- optical rotation
- refractive index
- R_f value
- ultraviolet spectrometry
- nature (colour, state, modifiers, description, *etc.*)

and represent what a typical organic chemist is expected to include in a report to show that they have made the intended substance.

The RSC provides guidelines on how each of the various analytical data should be displayed [83]. However, they do not insist that these are rigidly adhered to, which results in multiple (sometimes very similar) representations. For example, the suggested presentation of High Resolution Mass Spectroscopy data is:

> [Found: C, 63.1; H, 5.4%; M (mass spectrum), 352. $C_{13}H_{13}NO_4$ requires C, 63.2; H, 5.3%; M, 352]

Analysis of the corpus revealed the following representations:

- Calculated for $C_{13}H_8N_5Cl_3$: *m/z* 338.98. Found 338.98 HRMS
- (CI mode, CH_4): $C_{23}H_{24}Si$ (M); found: *m/z* 328.1647. Calc.: *m/z* 328.1647
- [Found: *m/z* (HRMS-FAB) 359.1993. $C_{21}H_{23}N_6$ requires MH^+ 359.1984]
- exact mass 292.0828, $C_{14}H_{14}NO_6$ ($M/2 + H^+$) requires 292.08
- (Found M^+, 257.9545. C_9H_7IO requires 257.9544)

- HR-LSIMS (*m*-nitrobenzyl alcohol) *m/z* 464.99649 [M(^{79}Br) + H$^+$], $C_{15}H_{18}{}^{79}BrN_2O_8S$ requires *m/z* 464.99673

- [MALDI-TOF-MS Calc. for $C_{127}H_{107}NNaO_{38}$ (M + Na)$^+$: 2276.6. Found: *m/z* 2276.9 (M+Na)$^+$]

- HRMS (CI$^+$) $C_{13}H_{16}BrO_2$ requires 283.0334, found 283.0329 (M + H)$^+$found = 423.1060, $C_{21}H_{25}ClO_5P$ requires 423.1128 for the ^{35}Cl isotope

- [M$^-$] *m/z* 801.0935. Calc. for [$C_{37}H_{33}Cl_3N_2O_{10}P$]$^-$: 801.0938

- ESI-MS *m/z*: 805.44 (M + H)$^+$;anal. calcd for $C_{52}H_{60}N_4S_2$: 804.43 *m/z* (EI) 414.1660 (M$^+$, 100%. $C_{23}H_{26}O_7$ requires 414.1678), 385 (16.8), 278 (7.1), 217 (6.2), 195 (34.8), 167 (86.5), 135 (66.0), 131 (10.5)

as well as the recommended version. The identification of each type of analytical data therefore requires various representations to be taken into account. Regular expressions provide exactly this functionality and are supported by Java which is a cross-platform language.

Java does not support the Microsoft Word document format but the text can be entered into a Java *text area* by using the *cut and paste* translation facility. This process results in the complete loss of formatting (other than new lines) and in some cases special character. For instance, the 'δ' character sometimes is translated as 'd' and in other cases is omitted entirely. An infrared spectrum such as;

ν_{max} (CHCl$_3$)/cm^{-1}2954, 2900, 1655 and 1603 (C=C), 876;

is typically translated to;

max (CHCl3)/cm-12954, 2900, 1655 and 1603 (C=C), 876;

Thus the translation into plain text introduces further variation of representation, further complicating the extraction process.

2.2.3 Regular Expressions

A regular expression is a *pattern* or *template* for matching a set of text strings [84]. The origins of regular expressions lie in mathematics and arose from finite state automata theory in the 1950s. The mathematical background of regular expressions is discussed in 3.1.6, however, they rapidly migrated into early text editors such as *qed* [85] where they are no longer *regular expressions* in the mathematical sense. Many different variations have occurred in both syntax and semantics; this work has concentrated on the regular expression package implemented by Java [86].

Regular expressions match a pattern against a subject (a string of characters); most characters in the regular expression stand for themselves. For example a straight-chain hydrocarbon, with up to ten carbon atoms and one optional multiple bond, would match against the following regular expression:

chain (locantgroup)? saturation

Where;

- chain is defined as (meth|eth|prop|but|pent|hex|hept|oct|non|dec)
- locantgroup is defined as -number-
- number is defined as (1|2|3|4|5|6|7|8|9)
- saturation is defined as (ane|ene|yne)

The vertical bar means *or*; the parentheses are used to group sub-expressions; the question mark means *zero or one instances of* and the juxtaposition of the parenthesised expressions means concatenation. Some common predefined character classes and boundary matches used in the Java regular expression package is given in appendix B.

It is clear that the regular expression given above would match against both possible and impossible chemical names, for example meth-9-yne. This limitation that requires that regular expressions only be used to tokenise data, not to deal with the semantics (see section 3.2).

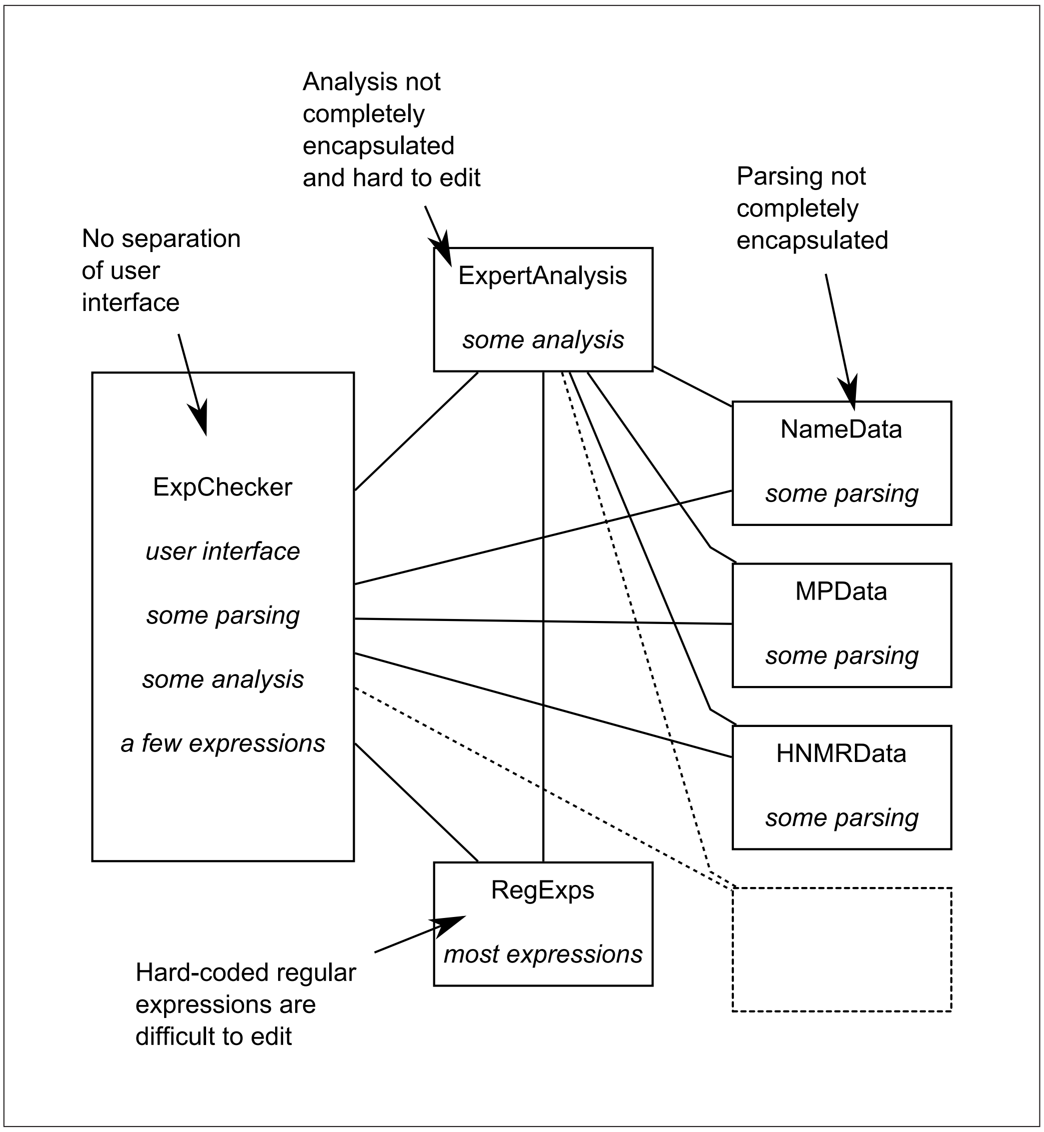
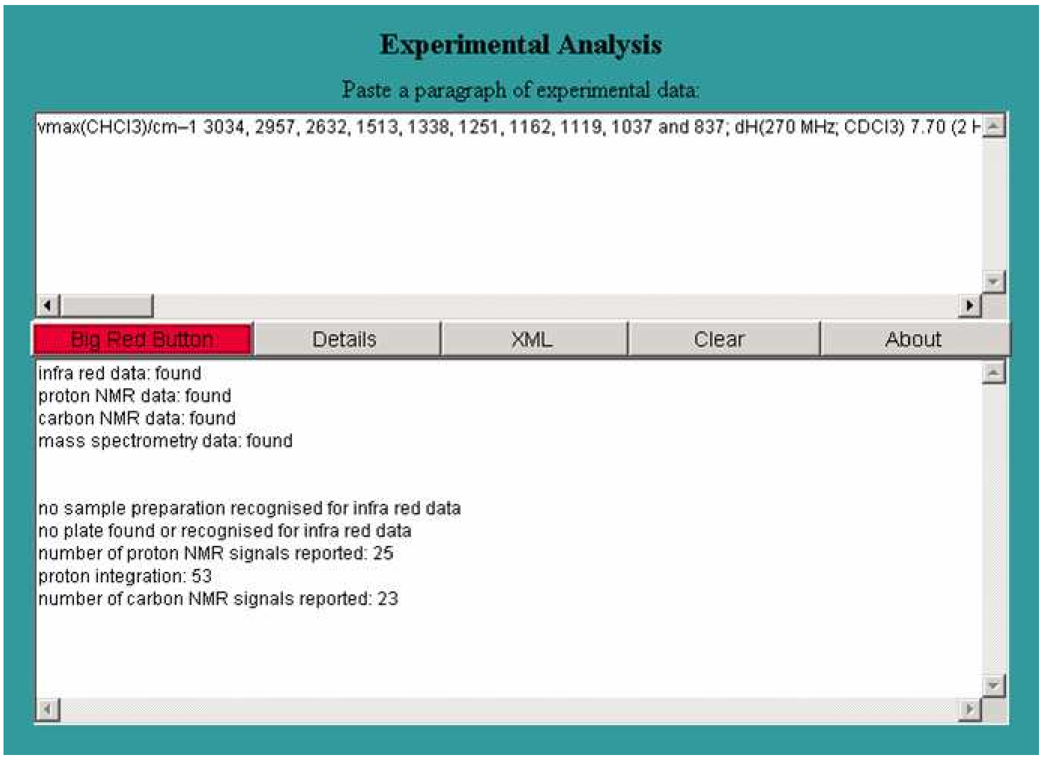


Figure 2.7: The structure of the original EDC.



```
vmax(CHCl3)/cm−1 3034, 2957, 2632, 1513, 1338, 1251, 1162, 1119, 1037 and 837; dH(270 MHz; CDCl3) 7.70 (2 H
```

```
infra red data: found
proton NMR data: found
carbon NMR data: found
mass spectrometry data: found


no sample preparation recognised for infra red data
no plate found or recognised for infra red data
number of proton NMR signals reported: 25
proton integration: 53
number of carbon NMR signals reported: 23
```

Figure 2.8: The GUI of the original EDC. An experimental paragraph has been pasted into the upper window and the big red button pressed to produce the report in the bottom window.

2.2.4 Structure of the EDC

The structure of the original EDC and the user interface are shown in figures 2.7 and 2.8 respectively. The two-panel display means that redundant input information (the unparsed text) is always displayed. Whilst this version of the EDC fulfilled the project requirements, there were significant weaknesses:

- the design of the code was poor: each of the types of data identified (melting point, name, HNMR *etc.*) were held in purpose built classes and did not utilise interfaces
- there was no separation of function and display
- each paragraph of the experimental section had to be individually cut and pasted
- the program was only available as an applet
- editing or extending the regular expressions required editing the Java source code

2.3 OSCAR — Blurring the Line Between Authoring Tool and IE Tool

The original EDC sufficed to show proof of concept. However, it had weaknesses and areas where the implementation and functionality could be improved. The RSC sponsored three further students (S. E. Adams, C. A. Waudby and this author) to continue the development of the EDC. The following areas were those were identified as those most in need of refactoring:

- separation of function and display
- analysis of entire article at once
- overview of entire article
- creation of an application

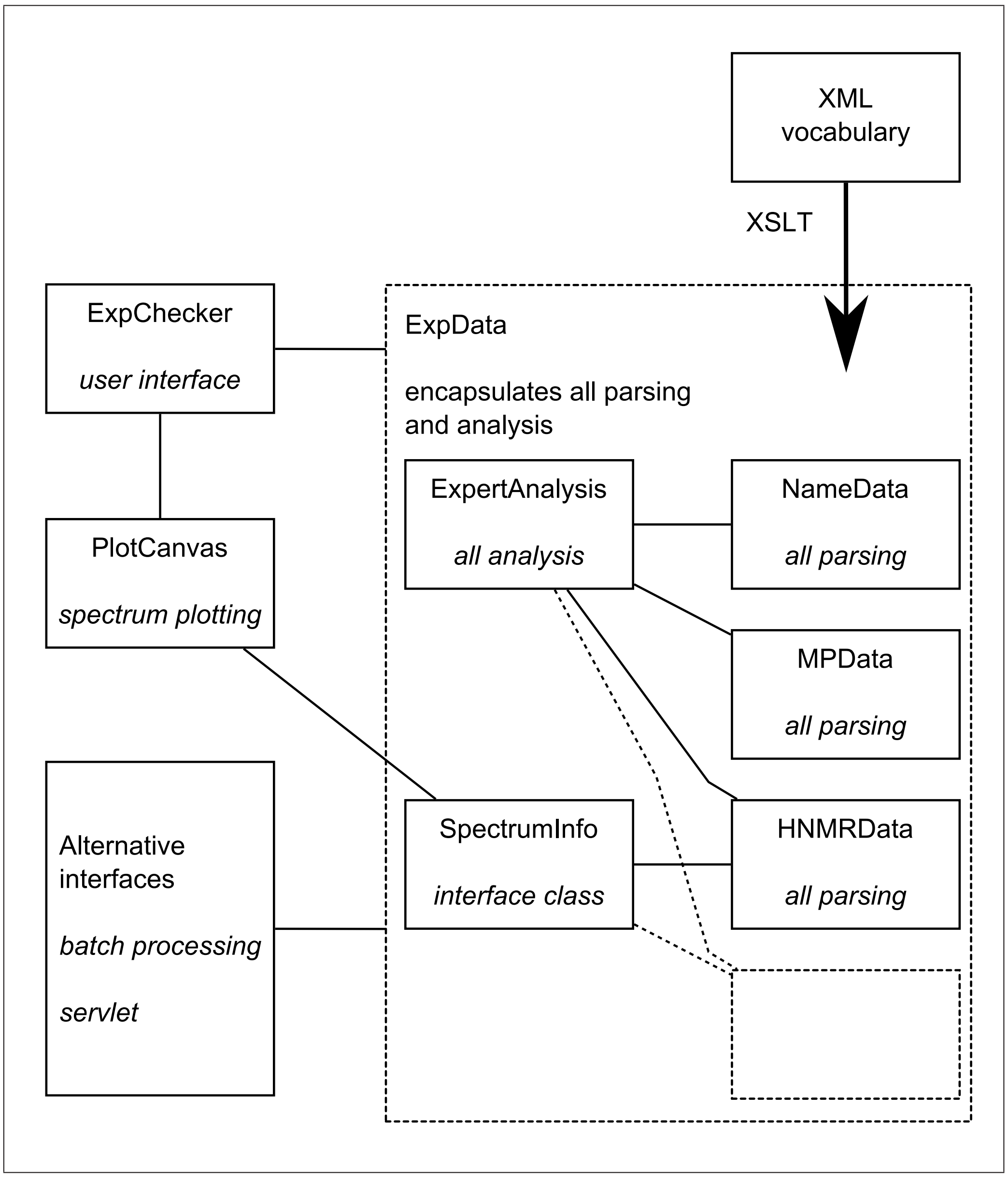


Figure 2.9: The structure of the EDC version 2.

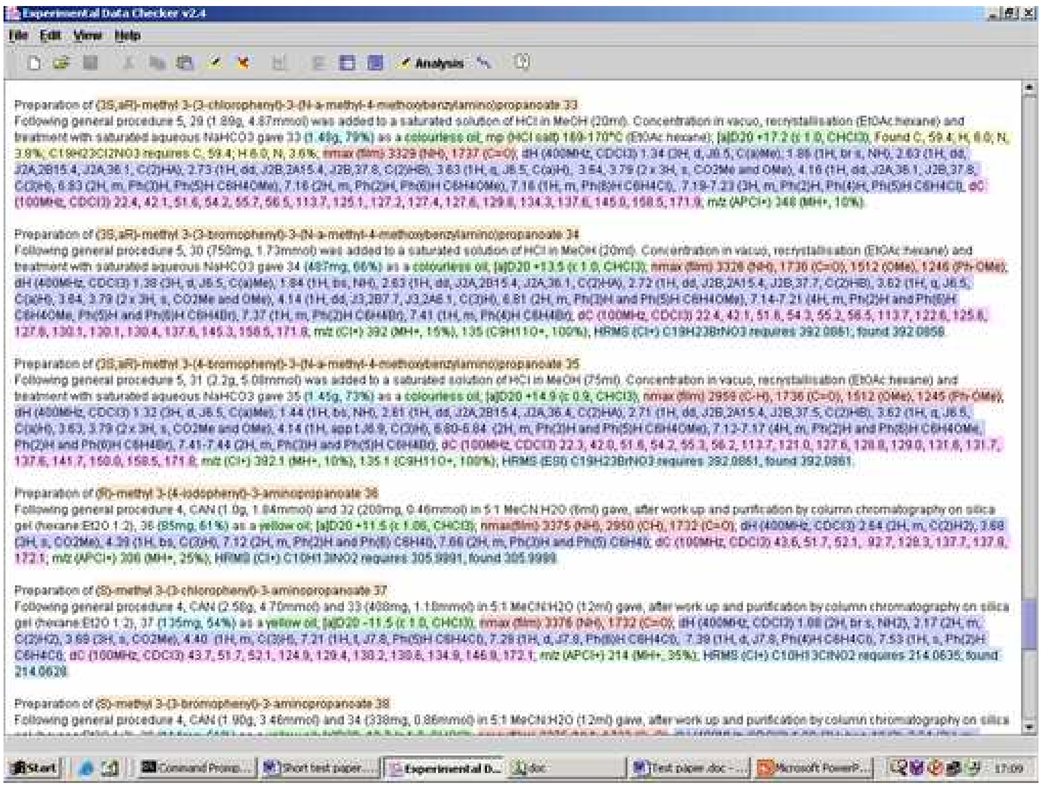
```
(\(|\[)?(F|f)ound\\s\(%\):.*(C|c)alcd?\.?\s for.*\d{1,2}\.\d{1,2}(\]\.|.;)

[\[\(]?(((?:anal\.?\s+)?found|(?:(?:(?:(?=\w{4,})\b(?x-i:(?:Zr|Zn|Yb|Y|Xe|W|V|Uuu|
Uut|Uus|Uuq|Uup|Uuo|Uuh|Uub|U|Tm|Tl|Ti|Th|Te|Tc|Tb|Ta|Sr|Sn|Sm|Si|Sg|Se|Sc|Sb|S|Ru|Rn|Rh|Rf|Re|Rb|Ra|
Pu|Pt|Pr|Po|Pm|Pd|Pb|Pa|P|Os|O|Np|No|Ni|Ne|Nd|Nb|Na|N|Mt|Mo|Mn|Mg|Md|Lu|Lr|Li|La|Kr|K|Ir|In|I|Hs|Ho|
Hg|Hf|He|H|Ge|Gd|Ga|Fr|Fm|Fe|F|Eu|Es|Er|Dy|Ds|Db|Cu|Cs|Cr|Co|Cm|Cl|Cf|Ce|Cd|Ca|C|Br|Bk|Bi|Bh|Be|Ba|
B|Au|At|As|Ar|Am|Al|Ag|Ac|Me|Et|Pr|Ph|Bn|Bz|Bu|t-?Bu|n-?Bu|Ts|Ms|Tr|D)\d*[--\-\=]?)+
(?:\s*[·.]\s*(?:\d+(?:\/\d+)?\s*)?(?x-i:[A-Z][a-z]?\d*)+)?\b\s*)?
require[sd]?\s*|(?:anal[\.:\s]+)?calcd?\.?(?:\s+for)?\W*(?:(?=\w{4,})\b
(?x-i:(?:Zr|Zn|Yb|Y|Xe|W|V|Uuu|Uut|Uus|Uuq|Uup|Uuo|Uuh|Uub|U|Tm|Tl|Ti|Th|Te|Tc|Tb|Ta|Sr|Sn|Sm|Si|Sg|
Se|Sc|Sb|S|Ru|Rn|Rh|Rf|Re|Rb|Ra|Pu|Pt|Pr|Po|Pm|Pd|Pb|Pa|P|Os|O|Np|No|Ni|Ne|Nd|Nb|Na|N|Mt|Mo|Mn|
Mg|Md|Lu|Lr|Li|La|Kr|K|Ir|In|I|Hs|Ho|Hg|Hf|He|H|Ge|Gd|Ga|Fr|Fm|Fe|F|Eu|Es|Er|Dy|Ds|Db|Cu|Cs|Cr|
Co|Cm|Cl|Cf|Ce|Cd|Ca|C|Br|Bk|Bi|Bh|Be|Ba|B|Au|At|As|Ar|Am|Al|Ag|Ac|Me|Et|Pr|Ph|Bn|Bz|Bu|t-?Bu|n-?Bu|
Ts|Ms|Tr|D)\d*[--\-\=]?)+(?:\s*[·.]\s*(?:\d+(?:\/\d+)?\s*)?(?x-i:[A-Z][a-z]?
\d*)+)?\b\s*)?(?:(?:\((?:[^\(\)]|\((?:[^\(\)]|\([^\(\)]+\))+\))+\))\s*)?|
for\W*)))(?:\W*(?:M\w*)?\W+)?((?:\W*?(?x-i:[A-Z][a-z]?)[\s,:]+\d+(?:\.\d+)?\s*
\%??){2,})%?[;\.\s]*?){2}[\)\]]?
```

Figure 2.10: Regular expressions to extract the elemental analysis from the analytical section. The top expression is that used in the EDC, bottom is that used in OSCAR.

- improve spectrum representation
- improve the regular expressions

A complete rewrite of the code produced the revised program design shown in figure 2.9. Each of the types of data now implement a `DataInterface` and those data representing spectra a `SpectrumInterface`. The separation of function and display resulted in OSCAR (Open Source Chemistry Analysis Routines) and the EDC which is now simply a particular *front end* for a user to access OSCAR [87]. Figures 2.11 and 2.12 show the EDC application highlighting data and the results of the 'expert analysis' routines. An example of the regular expressions developed for the original EDC and the improved version for OSCAR are shown in figure 2.10.

In order that an entire article could be parsed at once, the document structure and paragraph recognition are important. OSCAR firstly identifies the experimental section of the article and then splits this *ReportParagraphs*, where a ReportParagraph is defined in Backus-Naur form (BNF) in figure



Preparation of (3S,aR)-methyl 3-(3-chlorophenyl)-3-(N-a-methyl-4-methoxybenzylamino)propanoate 33
Following general procedure 5, 29 (1.89g, 4.87mmol) was added to a saturated solution of HCl in MeOH (20ml). Concentration in vacuo, recrystallisation (EtOAc:hexane) and treatment with saturated aqueous NaHCO3 gave 33 (1.48g, 79%) as a colourless oil; mp (HCl salt) 169-170°C (EtOAc:hexane); [a]D20 +17.2 (c 1.0, CHCl3); Found C, 59.4; H, 6.0; N, 3.9%; C19H23Cl2NO3 requires C, 59.4; H 6.0, N, 3.6%; nmax (film) 3329 (NH), 1737 (C=O); dH (400MHz, CDCl3) 1.34 (3H, d, J6.5, C(a)Me), 1.86 (1H, br s, NH), 2.63 (1H, dd, J2A,2B15.4, J2A,36.1, C(2)HA), 2.73 (1H, dd, J2B,2A15.4, J2B,37.8, C(2)HB), 3.63 (1H, q, J6.5, C(a)H), 3.64, 3.79 (2 x 3H, s, CO2Me and OMe), 4.16 (1H, dd, J2A,36.1, J2B,37.8, C(3)H), 6.83 (2H, m, Ph(3)H, Ph(5)H C6H4OMe), 7.16 (2H, m, Ph(2)H, Ph(6)H C6H4OMe), 7.16 (1H, m, Ph(6)H C6H4Cl), 7.19-7.23 (3H, m, Ph(2)H, Ph(4)H, Ph(5)H C6H4Cl); dC (100MHz, CDCl3) 22.4, 42.1, 51.6, 54.2, 55.7, 56.5, 113.7, 125.1, 127.2, 127.4, 127.6, 129.8, 134.3, 137.6, 145.0, 158.5, 171.9; m/z (APCI+) 348 (MH+, 10%).

Preparation of (3S,aR)-methyl 3-(3-bromophenyl)-3-(N-a-methyl-4-methoxybenzylamino)propanoate 34
Following general procedure 5, 30 (750mg, 1.73mmol) was added to a saturated solution of HCl in MeOH (20ml). Concentration in vacuo, recrystallisation (EtOAc:hexane) and treatment with saturated aqueous NaHCO3 gave 34 (487mg, 66%) as a colourless oil; [a]D20 +13.5 (c 1.0, CHCl3); nmax (film) 3326 (NH), 1736 (C=O), 1512 (OMe), 1246 (Ph-OMe); dH (400MHz, CDCl3) 1.38 (3H, d, J6.5, C(a)Me), 1.84 (1H, bs, NH), 2.63 (1H, dd, J2A,2B15.4, J2A,36.1, C(2)HA), 2.72 (1H, dd, J2B,2A15.4, J2B,37.7, C(2)HB), 3.62 (1H, q, J6.5, C(a)H), 3.64, 3.79 (2 x 3H, s, CO2Me and OMe), 4.14 (1H, dd, J3,2B7.7, J3,2A6.1, C(3)H), 6.81 (2H, m, Ph(3)H and Ph(5)H C6H4OMe), 7.14-7.21 (4H, m, Ph(2)H and Ph(6)H C6H4OMe, Ph(5)H and Ph(6)H C6H4Br), 7.37 (1H, m, Ph(2)H C6H4Br), 7.41 (1H, m, Ph(4)H C6H4Br); dC (100MHz, CDCl3) 22.4, 42.1, 51.6, 54.3, 55.2, 56.5, 113.7, 122.6, 125.6, 127.9, 130.1, 130.1, 130.4, 137.6, 145.3, 158.5, 171.9; m/z (CI+) 392 (MH+, 15%), 135 (C9H11O+, 100%); HRMS (CI+) C19H23BrNO3 requires 392.0861; found 392.0859.

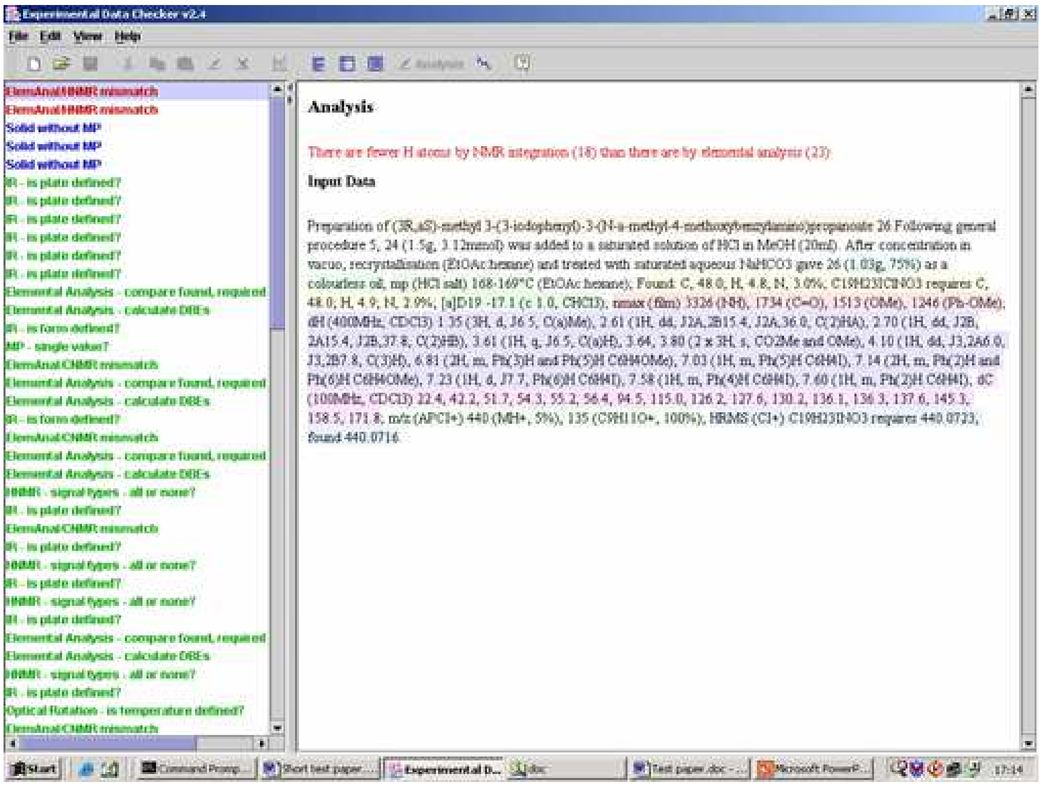
Preparation of (3S,aR)-methyl 3-(4-bromophenyl)-3-(N-a-methyl-4-methoxybenzylamino)propanoate 35
Following general procedure 5, 31 (2.2g, 5.08mmol) was added to a saturated solution of HCl in MeOH (75ml). Concentration in vacuo, recrystallisation (EtOAc:hexane) and treatment with saturated aqueous NaHCO3 gave 35 (1.45g, 73%) as a colourless oil; [a]D20 +14.9 (c 0.9, CHCl3); nmax (film) 2959 (C-H), 1736 (C=O), 1512 (OMe), 1245 (Ph-OMe); dH (400MHz, CDCl3) 1.32 (3H, d, J6.5, C(a)Me), 1.44 (1H, bs, NH), 2.61 (1H, dd, J2A,2B15.4, J2A,36.4, C(2)HA), 2.71 (1H, dd, J2B,2A15.4, J2B,37.5, C(2)HB), 3.62 (1H, q, J6.5, C(a)H), 3.63, 3.79 (2 x 3H, s, CO2Me and OMe), 4.14 (1H, app t J6.9, C(3)H), 6.80-6.84 (2H, m, Ph(3)H and Ph(5)H C6H4OMe), 7.13-7.17 (4H, m, Ph(2)H and Ph(6)H C6H4OMe, Ph(2)H and Ph(6)H C6H4Br), 7.41-7.44 (2H, m, Ph(3)H and Ph(5)H C6H4Br); dC (100MHz, CDCl3) 22.3, 42.0, 51.6, 54.2, 55.3, 56.2, 113.7, 121.0, 127.6, 128.8, 129.0, 131.6, 131.7, 137.6, 141.7, 150.0, 158.5, 171.8; m/z (CI+) 392.1 (MH+, 10%), 135.1 (C9H11O+, 100%); HRMS (ESI) C19H23BrNO3 requires 392.0861, found 392.0861.

Preparation of (R)-methyl 3-(4-iodophenyl)-3-aminopropanoate 36
Following general procedure 4, CAN (1.0g, 1.84mmol) and 32 (200mg, 0.46mmol) in 5:1 MeCN:H2O (6ml) gave, after work up and purification by column chromatography on silica gel (hexane:Et2O 1:2), 36 (85mg, 61%) as a yellow oil; [a]D20 +11.5 (c 1.06, CHCl3); nmax(film) 3375 (NH), 2950 (CH), 1732 (C=O); dH (400MHz, CDCl3) 2.64 (2H, m, C(2)H2), 3.68 (3H, s, CO2Me), 4.39 (1H, bs, C(3)H), 7.12 (2H, m, Ph(2)H and Ph(6) C6H4I), 7.66 (2H, m, Ph(3)H and Ph(5) C6H4I); dC (100MHz, CDCl3) 43.6, 51.7, 52.1, 92.7, 128.3, 137.7, 137.8, 172.1; m/z (APCI+) 306 (MH+, 25%); HRMS (CI+) C10H13INO2 requires 305.9991, found 305.9999.

Preparation of (S)-methyl 3-(3-chlorophenyl)-3-aminopropanoate 37
Following general procedure 4, CAN (2.58g, 4.70mmol) and 33 (408mg, 1.18mmol) in 5:1 MeCN:H2O (12ml) gave, after work up and purification by column chromatography on silica gel (hexane:Et2O 1:2), 37 (135mg, 54%) as a yellow oil; [a]D20 -11.5 (c 1.0, CHCl3); nmax (film) 3376 (NH), 1732 (C=O); dH (400MHz, CDCl3) 1.88 (2H, br s, NH2), 2.17 (2H, m, C(2)H2), 3.69 (3H, s, CO2Me), 4.40 (1H, m, C(3)H), 7.21 (1H, t, J7.8, Ph(5)H C6H4Cl), 7.29 (1H, d, J7.8, Ph(6)H C6H4Cl), 7.39 (1H, d, J7.8, Ph(4)H C6H4Cl), 7.53 (1H, s, Ph(2)H C6H4Cl); dC (100MHz, CDCl3) 43.7, 51.7, 52.1, 124.9, 129.4, 130.2, 130.6, 134.9, 146.9, 172.1; m/z (APCI+) 214 (MH+, 35%); HRMS (CI+) C10H13ClNO2 requires 214.0635; found 214.0629.

Preparation of (S)-methyl 3-(3-bromophenyl)-3-aminopropanoate 38
Following general procedure 4, CAN (1.90g, 3.46mmol) and 34 (338mg, 0.86mmol) in 5:1 MeCN:H2O (12ml) gave, after work up and purification by column chromatography on silica

Figure 2.11: The EDC application v2.4 showing the data identified.



```
ElemAnal/HNMR mismatch
ElemAnal/HNMR mismatch
Solid without MP
Solid without MP
Solid without MP
IR - is plate defined?
IR - is plate defined?
IR - is plate defined?
IR - is plate defined?
IR - is plate defined?
IR - is plate defined?
Elemental Analysis - compare found, required
Elemental Analysis - calculate DBEs
IR - is form defined?
MP - single value?
ElemAnal/CNMR mismatch
Elemental Analysis - compare found, required
Elemental Analysis - calculate DBEs
IR - is form defined?
ElemAnal/CNMR mismatch
Elemental Analysis - compare found, required
Elemental Analysis - calculate DBEs
HNMR - signal types - all or none?
IR - is plate defined?
ElemAnal/CNMR mismatch
IR - is plate defined?
HNMR - signal types - all or none?
IR - is plate defined?
HNMR - signal types - all or none?
IR - is plate defined?
Elemental Analysis - compare found, required
Elemental Analysis - calculate DBEs
HNMR - signal types - all or none?
IR - is plate defined?
Optical Rotation - is temperature defined?
ElemAnal/CNMR mismatch
```

Analysis

There are fewer H atoms by NMR integration (18) than there are by elemental analysis (23)

Input Data

Preparation of (3R,aS)-methyl 3-(3-iodophenyl)-3-(N-a-methyl-4-methoxybenzylamino)propanoate 26 Following general procedure 5, 24 (1.5g, 3.12mmol) was added to a saturated solution of HCl in MeOH (20ml). After concentration in vacuo, recrystallisation (EtOAc:hexane) and treated with saturated aqueous NaHCO3 gave 26 (1.03g, 75%) as a colourless oil, mp (HCl salt) 168-169°C (EtOAc:hexane); Found: C, 48.0; H, 4.8; N, 3.0%; C19H23ClINO3 requires C, 48.0; H, 4.9; N, 2.9%; [a]D19 -17.1 (c 1.0, CHCl3); nmax (film) 3326 (NH), 1734 (C=O), 1513 (OMe), 1246 (Ph-OMe); dH (400MHz, CDCl3) 1.35 (3H, d, J6.5, C(a)Me), 2.61 (1H, dd, J2A,2B15.4, J2A,36.0, C(2)HA), 2.70 (1H, dd, J2B,2A15.4, J2B,37.8, C(2)HB), 3.61 (1H, q, J6.5, C(a)H), 3.64, 3.80 (2 x 3H, s, CO2Me and OMe), 4.10 (1H, dd, J3,2A6.0, J3,2B7.8, C(3)H), 6.81 (2H, m, Ph(3)H and Ph(5)H C6H4OMe), 7.03 (1H, m, Ph(5)H C6H4I), 7.14 (2H, m, Ph(2)H and Ph(6)H C6H4OMe), 7.23 (1H, d, J7.7, Ph(6)H C6H4I), 7.58 (1H, m, Ph(4)H C6H4I), 7.60 (1H, m, Ph(2)H C6H4I), dC (100MHz, CDCl3) 22.4, 42.2, 51.7, 54.3, 55.2, 56.4, 94.5, 115.0, 126.2, 127.6, 130.2, 136.1, 136.3, 137.6, 145.3, 158.5, 171.8; m/z (APCI+) 440 (MH+, 5%), 135 (C9H11O+, 100%); HRMS (CI+) C19H23INO3 requires 440.0723, found 440.0716

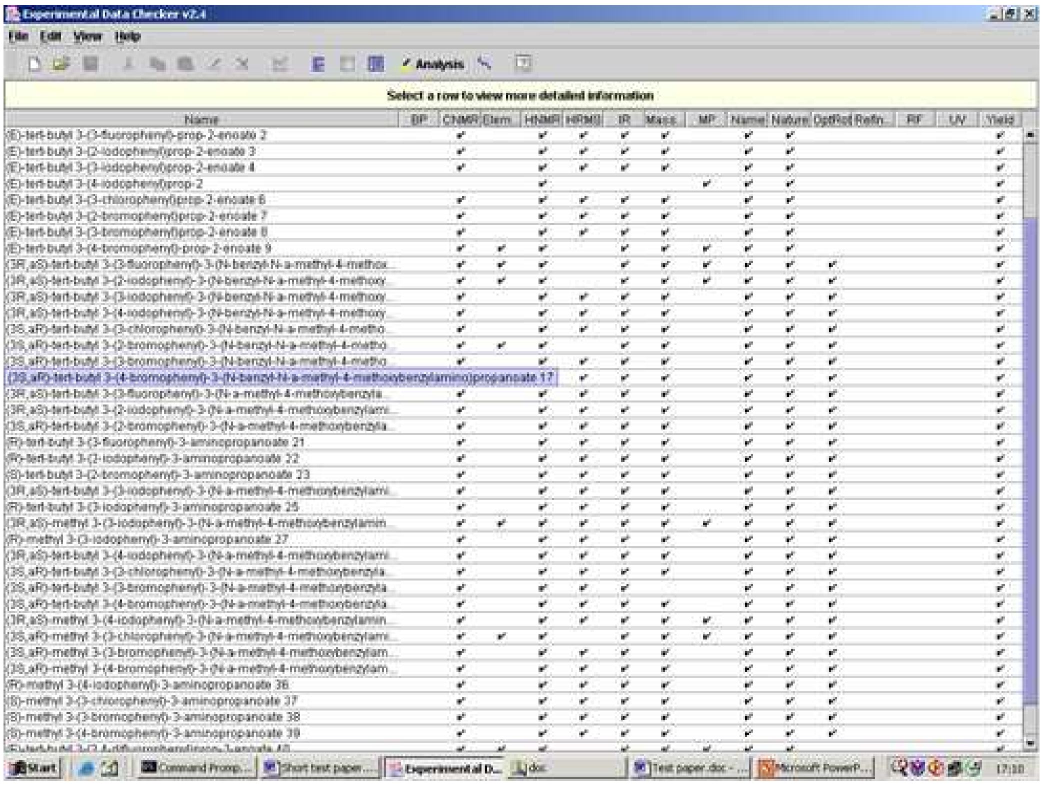
Figure 2.12: The EDC application v2.4 showing the results of the expert analysis routines. Warnings are in the left hand panel — clicking on a warning brings the relevant explanation up in the right hand column. Items in red represent serious errors (for example more Hydrogen atoms reported in the HNMR than there are in the elemental analysis. Blue items are warnings (for example solids being reported without melting points) and green possible warnings (for example an infra-red spectrum being reported without the plate type).

```
ReportParagraph ::  = ReportParagraphHeader AnalyticalDataBlock
                    FULLSTOP ReportParagraphEnd
                    ;

ReportParagraphHeader ::  = ChemicalName
                          | ChemicalNameAndIdentifier
                          | Identifier
                          ;

ReportParagraphEnd ::  = WhiteSpaces
                       | MaybeWhiteSpaces LINEEND
                       ;

WhiteSpaces ::  = WHITESPACE
                | WhiteSpaces WHITESPACE
                ;

MaybeWhiteSpaces ::  = BLANK
                     | WhiteSpaces
                     ;
```

Figure 2.13: The BNF for paragraph recognition in OBC papers. Terminal tokens are shown in capitals.

2.13 (for a full explanation of BNF see appendix C). Each of these ReportParagraphs is then examined for analytical data. The ability to parse an entire article at once — rather than individual paragraphs — allows the creation of a table listing all the compounds identified and the analytical data for each. This facility is considered particularly useful for chemists authoring a thesis in the organic chemistry, as a way to ensure that they have included all the required data [88]. Figure 2.14 shows an example of the tabulation facility.

The original EDC used regular expressions to identify fine-grained data immediately (for example matching the HNMR peak types from an entire paragraph). Such immediate fine-grained parsing is difficult because it is not easy to differentiate between HNMR and CNMR peak types, for example; thus the regular expressions are extremely complex. The approach in OSCAR was to use hierarchical parsing (not coincidentally reflecting the structure of the target XML output). With reference to HNMR this process would involve



Select a row to view more detailed information

Name	BP	CNMR	Elem	HNMR	HRMS	IR	Mass	MP	Name	Nature	OptRot	Refn...	RF	UV	Yield
(E)-tert-butyl 3-(3-fluorophenyl)-prop-2-enoate 2		✓		✓	✓	✓	✓		✓	✓					✓
(E)-tert-butyl 3-(2-iodophenyl)prop-2-enoate 3		✓		✓	✓	✓	✓		✓	✓					✓
(E)-tert-butyl 3-(3-iodophenyl)prop-2-enoate 4		✓		✓	✓	✓	✓		✓	✓					✓
(E)-tert-butyl 3-(4-iodophenyl)prop-2				✓				✓	✓	✓					✓
(E)-tert-butyl 3-(3-chlorophenyl)prop-2-enoate 6		✓		✓	✓	✓	✓		✓	✓					✓
(E)-tert-butyl 3-(2-bromophenyl)prop-2-enoate 7		✓		✓	✓	✓	✓		✓	✓					✓
(E)-tert-butyl 3-(3-bromophenyl)prop-2-enoate 8		✓		✓	✓	✓	✓		✓	✓					✓
(E)-tert-butyl 3-(4-bromophenyl)-prop-2-enoate 9		✓	✓	✓		✓	✓	✓	✓	✓					✓
(3R,aS)-tert-butyl 3-(3-fluorophenyl)-3-(N-benzyl-N-a-methyl-4-methox...		✓	✓	✓		✓	✓	✓	✓	✓	✓				✓
(3R,aS)-tert-butyl 3-(2-iodophenyl)-3-(N-benzyl-N-a-methyl-4-methoxy...		✓	✓	✓		✓	✓	✓	✓	✓	✓				✓
(3R,aS)-tert-butyl 3-(3-iodophenyl)-3-(N-benzyl-N-a-methyl-4-methoxy...		✓		✓	✓	✓	✓		✓	✓	✓				✓
(3R,aS)-tert-butyl 3-(4-iodophenyl)-3-(N-benzyl-N-a-methyl-4-methoxy...		✓		✓	✓	✓	✓		✓	✓	✓				✓
(3S,aR)-tert-butyl 3-(3-chlorophenyl)-3-(N-benzyl-N-a-methyl-4-metho...		✓		✓	✓	✓	✓		✓	✓	✓				✓
(3S,aR)-tert-butyl 3-(2-bromophenyl)-3-(N-benzyl-N-a-methyl-4-metho...		✓	✓	✓		✓	✓		✓	✓	✓				✓
(3S,aR)-tert-butyl 3-(3-bromophenyl)-3-(N-benzyl-N-a-methyl-4-metho...		✓		✓	✓	✓	✓		✓	✓	✓				✓
(3S,aR)-tert-butyl 3-(4-bromophenyl)-3-(N-benzyl-N-a-methyl-4-methoxybenzylamino)propanoate 17					✓	✓	✓		✓	✓	✓				✓
(3R,aS)-tert-butyl 3-(3-fluorophenyl)-3-(N-a-methyl-4-methoxybenzyla...		✓		✓	✓	✓	✓		✓	✓	✓				✓
(3R,aS)-tert-butyl 3-(2-iodophenyl)-3-(N-a-methyl-4-methoxybenzylami...		✓		✓	✓	✓	✓		✓	✓	✓				✓
(3S,aR)-tert-butyl 3-(2-bromophenyl)-3-(N-a-methyl-4-methoxybenzyla...		✓		✓	✓	✓	✓		✓	✓	✓				✓
(R)-tert-butyl 3-(3-fluorophenyl)-3-aminopropanoate 21		✓		✓	✓	✓	✓		✓	✓	✓				✓
(R)-tert-butyl 3-(2-iodophenyl)-3-aminopropanoate 22		✓		✓	✓	✓	✓		✓	✓	✓				✓
(S)-tert-butyl 3-(2-bromophenyl)-3-aminopropanoate 23		✓		✓	✓	✓	✓		✓	✓	✓				✓
(3R,aS)-tert-butyl 3-(3-iodophenyl)-3-(N-a-methyl-4-methoxybenzylami...		✓		✓	✓	✓	✓		✓	✓	✓				✓
(R)-tert-butyl 3-(3-iodophenyl)-3-aminopropanoate 25		✓		✓	✓	✓	✓		✓	✓	✓				✓
(3R,aS)-methyl 3-(3-iodophenyl)-3-(N-a-methyl-4-methoxybenzylamin...		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓				✓
(R)-methyl 3-(3-iodophenyl)-3-aminopropanoate 27		✓		✓	✓	✓	✓		✓	✓	✓				✓
(3R,aS)-tert-butyl 3-(4-iodophenyl)-3-(N-a-methyl-4-methoxybenzylami...		✓		✓	✓	✓	✓		✓	✓	✓				✓
(3S,aR)-tert-butyl 3-(3-chlorophenyl)-3-(N-a-methyl-4-methoxybenzyla...		✓		✓	✓	✓	✓		✓	✓	✓				✓
(3S,aR)-tert-butyl 3-(3-bromophenyl)-3-(N-a-methyl-4-methoxybenzyla...		✓		✓	✓	✓			✓	✓	✓				✓
(3S,aR)-tert-butyl 3-(4-bromophenyl)-3-(N-a-methyl-4-methoxybenzyla...		✓		✓	✓	✓	✓		✓	✓	✓				✓
(3R,aS)-methyl 3-(4-iodophenyl)-3-(N-a-methyl-4-methoxybenzylamin...		✓		✓	✓	✓	✓	✓	✓	✓	✓				✓
(3S,aR)-methyl 3-(3-chlorophenyl)-3-(N-a-methyl-4-methoxybenzylami...		✓	✓	✓		✓	✓	✓	✓	✓	✓				✓
(3S,aR)-methyl 3-(3-bromophenyl)-3-(N-a-methyl-4-methoxybenzylam...		✓		✓	✓	✓	✓		✓	✓	✓				✓
(3S,aR)-methyl 3-(4-bromophenyl)-3-(N-a-methyl-4-methoxybenzylam...		✓		✓	✓	✓	✓		✓	✓	✓				✓
(R)-methyl 3-(4-iodophenyl)-3-aminopropanoate 36		✓		✓	✓	✓	✓		✓	✓	✓				✓
(S)-methyl 3-(3-chlorophenyl)-3-aminopropanoate 37		✓		✓	✓	✓	✓		✓	✓	✓				✓
(S)-methyl 3-(3-bromophenyl)-3-aminopropanoate 38		✓		✓	✓	✓	✓		✓	✓	✓				✓
(S)-methyl 3-(4-bromophenyl)-3-aminopropanoate 39		✓		✓	✓	✓	✓		✓	✓	✓				✓
(E)-tert-butyl 3-(2,4-difluorophenyl)prop-2-enoate 40		✓	✓	✓		✓	✓	✓	✓	✓					✓

Figure 2.14: The EDC application v2.4 showing the tabulation of identified data. Clicking on a row brings up the relevant paragraph with more detailed information.

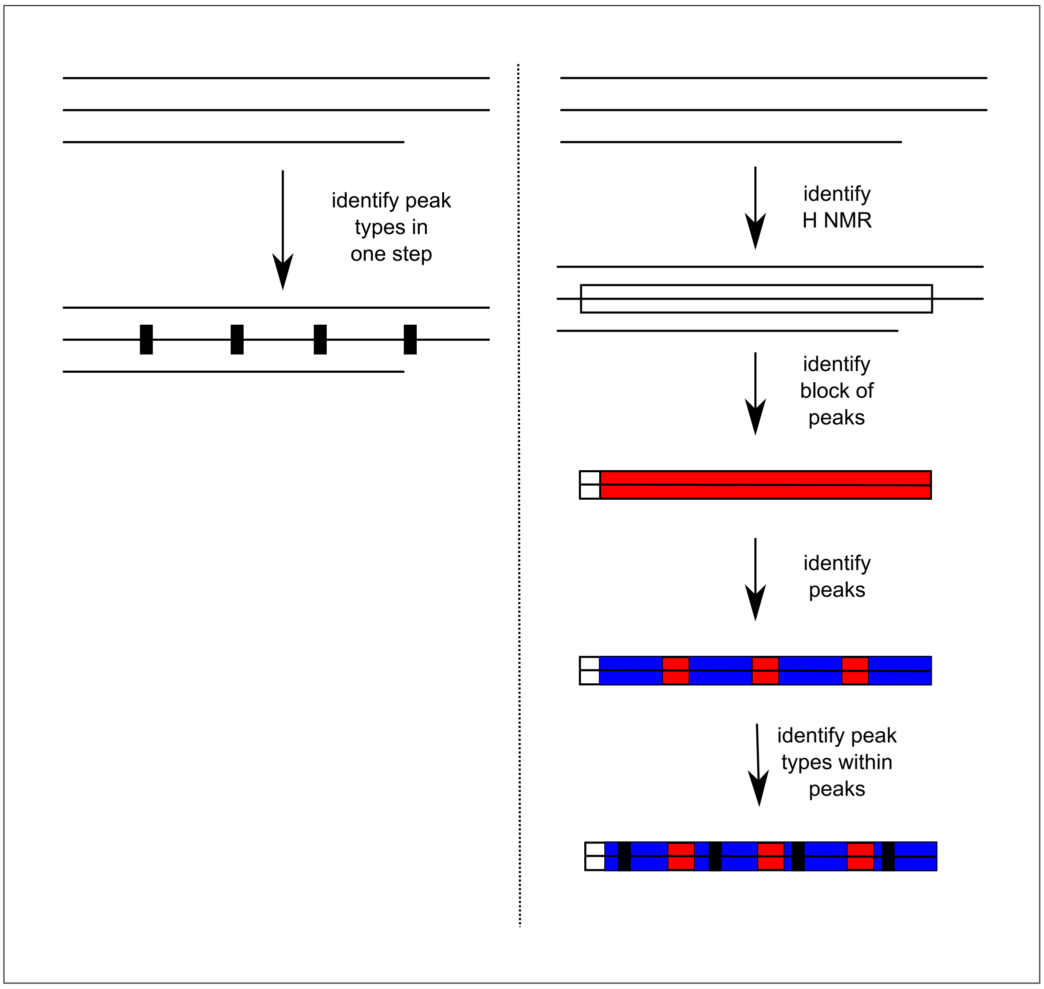

identify peak types in one step
identify H NMR
identify block of peaks
identify peaks
identify peak types within peaks

Figure 2.15: All-at-once (left) and hierarchical (right) parsing strategies.

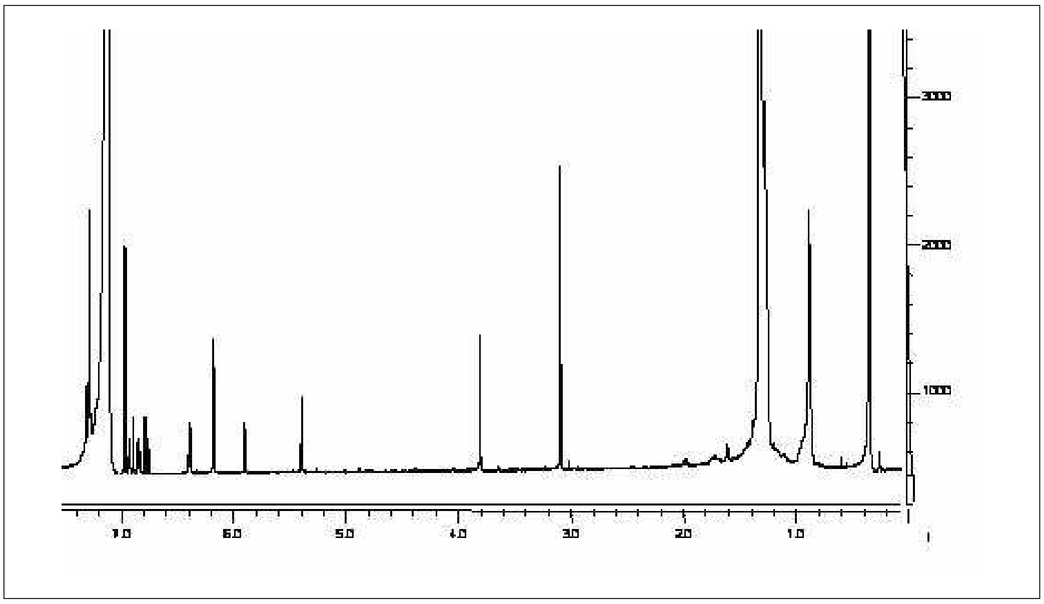


Figure 2.16: An example of an HNMR spectrum.

the following steps; identifying the block of HNMR data, then the block of peaks within that, then individual peaks and finally the peak type within the peak fragment. The two processes are illustrated in figure 2.15.

Organic chemists are familiar with graphical representations of spectra such as the HNMR spectrum shown in figure 2.16. However one of the drawback of the current publication system is the destruction of information; spectra are not given in a graphical form, but in an abbreviated textual representation as shown in figure 2.2. A system was therefore created that attempted to recreate a spectrum from the textual form. A mass spectrum is inherently easier to recreate than an infra-red spectrum as it consists of single lines of differing height whilst the infra-red spectrum has different width and shapes of peak. Figure 2.17 shows an example of a recreated spectrum.

2.3.1 Information Extraction Tests

In 2003 the RSC created a new journal, Organic & Biomolecular Chemistry (OBC), which was formed by the merger of the Journal of the Chemical

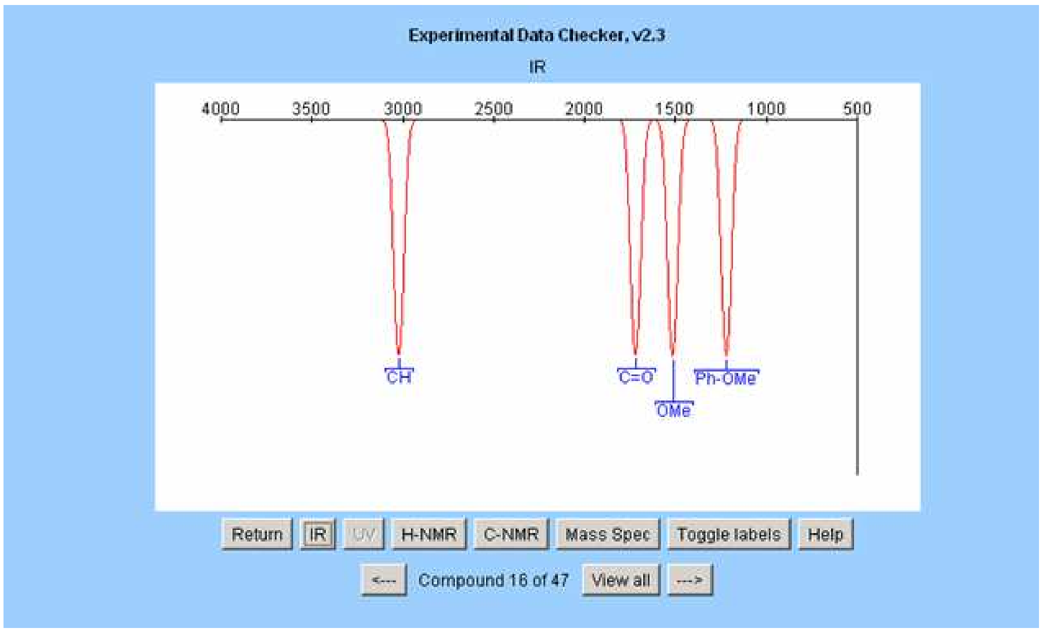


Figure 2.17: The EDC applet v2.3 displaying a recreated infra-red spectrum.

Society, Perkin Trans 1, Perkin Trans 2 and Natural Product Reports. Fortunately the regular expressions developed to identify and extract data from the original corpus proved sufficiently broad (and the structure of the analytical sections sufficiently rigid) that the desired data could still be extracted from the new journal. It is interesting to note that OSCAR was run over organic chemistry articles in German and still correctly identified typically more than 80% of the data correctly.

To determine the recall and accuracy rates for OSCAR it is necessary to consider the two types of document for which it is applicable: published OBC articles and papers received by the RSC for publication in OBC. There are only minor differences between the two document types, the two most relevant being structure (order) and format. The first of these should not present any parsing problems because although the experimental section may occur in a different position in the two documents, the content is the same in both. However, the format of the data often changes slightly in the publication process. In general slight errors (such as missing commas or mismatched brackets) found in the analytical sections of the submitted documents will be corrected before publication.

The EDC was originally intended to function as an authoring tool; thus it was decided at the inception of the project that the program should not make inferences from the data or correct it but that a human must perform curation functions. Therefore an error such as the mislabelling of a HNMR spectrum as a CNMR spectrum that would be clear to a chemist reading the paper because of the assignment of the integrals as protons and the values of the chemical shifts would be interpreted by OSCAR as a CNMR spectrum. Errors of this type are usually corrected during the review and editing process. Hence it was expected that the recall rate would be lower than the precision rate, (a single missing comma in a spectrum would result in a false negative) and that *tight* specifications[‡] for a data type would result in higher recall.

The test set comprised seven articles randomly selected from OBC 2003 [89, 90, 91, 92, 93, 94, 95] and three documents [96, 97, 98] that the RSC had received for submission to OBC. Because the focus for the current project is information extraction from published literature the test set was biased in favour of articles. The recall and accuracy statistics for the classification (identification) obtained from this sample are given in table 2.2, where:

TP is the number of X that the system correctly identified and were present in the corpus

FN is the number of X that the system failed to identify

FP is the number of X that were recognised by the system which were not in the corpus

X is a particular data type

[‡]A tight specification is one where there is little or no ambiguity possible.

Data type	*TP*	*FN*	*FP*	*Recall* %	*Precision* %
overall	1554	240	96	86.62	94.18
CNMR	187	24	5	93.03	97.40
elemental analysis	103	15	0	87.29	100.00
HNMR	212	23	4	90.21	98.15
HRMS	126	1	0	99.21	100.00
infra-red spectroscopy	186	19	8	90.73	95.88
mass spectroscopy	145	20	0	87.88	100.00
melting point	151	11	2	93.21	98.69
chemical name	171	72	47	70.37	78.44
nature	100	22	21	81.97	82.64
yield	173	43	9	80.09	95.05

Table 2.2: Recall and precision rates for OSCAR.

and recall and precision are defined by

$$Recall = \frac{TP}{TP + FN} \tag{2.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

A partially identified fragment has been classified as a false positive, hence missing commas or mismatched brackets in a spectrum would lead to a false positive. It is observed that the recall of the first seven data types is about 90%, but for the final three the recall is significantly lower. It is also noted that the precision for two of these data types (chemical name and nature) is also significantly lower than for the other types.

In general it was observed that the reason why the yield was not correctly identified when multiple yields were quoted in one synthetic paragraph was because more than one synthetic route to the same compound was attempted, or the reaction gave a mixture of products, or the same reaction scheme had been employed on multiple starting compounds (see figure 2.18).

Typical procedure for the synthesis of 2-alkoxy-9-benzyl-8-hydroxyadenine derivatives 7a-e
A solution of **12** (0.24 mmol) in *c.* HCl was stirred at room temperature for 4 h. After evaporation, the residue was chromatographed on silica gel to give **7**, which was identified by comparison with a standard sample synthesized from **6**. Yield: **7a** (74%); **7b** (81%); **7c** (82%); **7d** (87%); **7e** (84%). [90]

Figure 2.18: A paragraph showing how the synthesis of set of similar compounds is described.

The relatively low recall and precision rates for identification of the chemical names and nature is not surprising. For chemical names an extremely simple regular expression was used: a match for various starts of chemical names, various ways of separating names (such as hyphens) and a match for various ends of chemical names. The nature of a compound is the reported data type that has the least-controlled vocabulary and hence presents more problems to parse. Although it would be possible to build up larger lists to match the state, colour and colour-modifiers of a compound that would increase the recall, an entirely different approach such as entity recognition would be preferable.

The semantic and syntactic features that caused the majority of the false negatives were:

- multiple instances of the same data type reported in the same paragraph
- inability to recognise the end of the reported data for a compound

The first of these would be simple to correct: OSCAR was designed to search for only one instance of a particular data type in a particular paragraph, however the inclusion of a recursive descent call would allow it to identify multiple instances of the same data type. The second is far more difficult because the *chunking* of the article into report-paragraphs (see above) is not sufficiently accurate.

	General		Experimental	
γ	Precision %	Recall %	Precision %	Recall %
-2	68.5	94.3	85.9	95.3
-5	72.1	92.5	93.8	95.3
-8	75.0	90.6	96.8	95.4
-11	**81.0**	**88.7**	**98.4**	**95.3**
-14	80.8	79.2	98.2	87.5

Table 2.3: The effect of the γ parameter on the recall and precision rates of OSCAR2 for chemical names in the general and experimental sections. The values in bold font are those representing the optimal value chosen [99].

There are two inherent weaknesses to the method of paragraph recognition used by OSCAR. The first occurs if the author omitted a full stop at the end of one analytical report section, which means that all the subsequent ReportParagraphs will be passed over until a full stop followed by optional whitespace then a newline is encountered. This produces significant numbers of false negatives and also leads to any previously un-encountered data types present in the subsequent paragraphs being reported for the incorrect compound. The second weakness is that this construction may also match sections earlier in the experimental section, giving rise to false positives.

2.4 OSCAR2 — the Importance of Chemical Names

Waudby continued the development of the OSCAR toolkit, focusing on improvements to chemical name recognition and maintaining more of the article structure[§]. The version Waudby developed produced documents with inline XML and separated the chemical name recognition from analytical data identification. The advantage of using inline XML markup is that it allows the preservation of the original document structure.

A naïve Baysian based on n-grams and a simple grammar (see section 3.1) were implemented to determine whether a word, or phrase was likely to be

[§]OSCAR2 is entirely the work of Waudby.

a chemical name. Briefly, this involved breaking words up into three or four letter tokens and determining which tokens occur more in chemicals. For example, *ybd*, *eth*, *alk*, *yne* might all be expected to occur more frequently in chemical names than in general English. The grammar allows the context of the word to be taken into account. For example, a word is likely to be a chemical if it is followed by a quantity:

methanol (10cm^3)

Single compounds with space-separated chemical names can also be correctly identified by context. For example, if a word ends in 'ic' is it followed by 'acid':

periodic acid, periodic table

The first instance of 'periodic' is followed by acid and is therefore assumed to be a chemical name whereas the second is followed by 'table' so is not identified as a chemical name. The system included a variable parameter γ that could be tuned to adjust the balance between recall and precision. Table 2.3 shows the recall and precision rates for five values of this parameter. A more complete explanation of the approach can be found in the article by Townsend *et al.* [100] and the various approaches this has built on in the articles by Vasserman [101]. Figure 2.19 shows the workflow for the OSCAR2 toolkit.

2.4.1 The Importance of Connection Tables

Chemical structures form the basis of most organic chemistry; the two dimensional representations of molecules (sometimes with an indication of the three dimensional structure included) are the form which chemists use when describing a molecule, or a reaction mechanism. The structure of an organic molecule, whether two or three dimensional, can usually be described by a connection table (CT). Systematic chemical names are often avoided (until the molecule is to be included in a formal report). In common use they are often in abbreviated forms or non-systematic names because systematic names are often lengthy, difficult to interpret and less memorable (see figure 2.6).

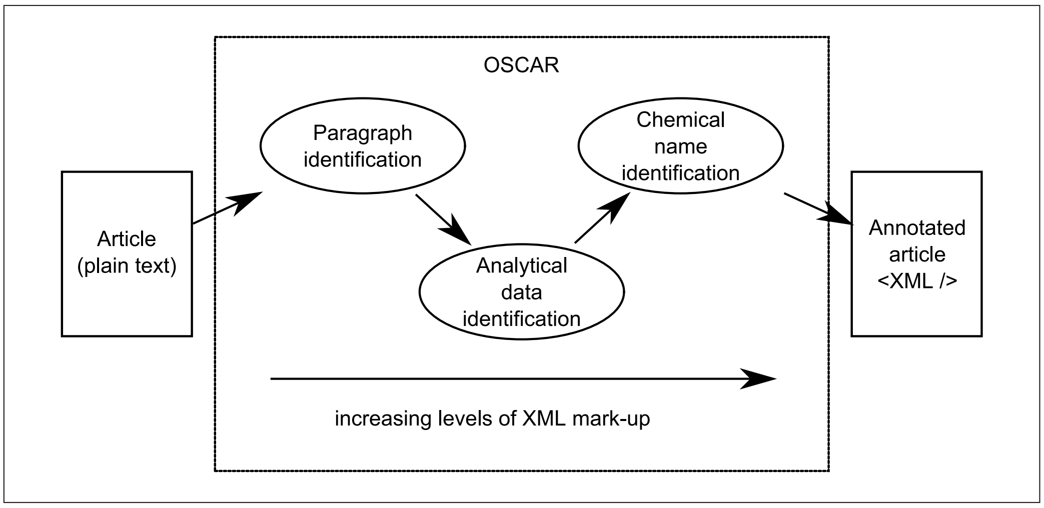

OSCAR
Paragraph identification
Chemical name identification
Article (plain text)
Analytical data identification
Annotated article <XML />
increasing levels of XML mark-up

Figure 2.19: The OSCAR workflow.

The identification of chemical names is vital because it is often the only way that the CT of the molecule can be recovered. The CT forms the basis of most organic chemistry, whether it be in predicting a reaction pathway or determining the likely appearance of infra-red, HNMR or CNMR spectra. Data of this type is frequently reported to characterise the molecule. It would be desirable to use all available data reported in a paper to perform crosschecks and self-validation. Whilst electronic publications provide the means to encapsulate a compound's CT in a document, this is most commonly achieved by including an MDL molfile [44] or a ChemDraw file [102] into a Microsoft Word document (which are rendered to the desired form when the file is viewed). During the publication process the CT is often converted into a picture, with the associated loss of data, leaving only the name available as a basis from which to recreate the CT. This is also the case with older articles where there is no electronic form in existence.

There are good commercial chemical name-to-structure converters available (for example ChemDraw). However, their analysis techniques are not published but it is believed that the algorithms are rule-based with no machine-learning capability. It is instructive to consider why a new approach to chemical name-to-structure conversion is useful and why a technique incorporating some machine-learning aspects should be employed.

2.5 OSCAR3 and OPSIN — Parsing Chemical Names to CTs

It is now common to use programs to generate the systematic name for a compound. This has resulted in improved compliance to the IUPAC specifications and greatly increased quality in the reported names. As computers are used to generate the names it seems sensible also also use them to perform the reverse translation.

There have been a significant number of attempts to produce automated methods to convert a chemical name to a CT. The first use of a computerised grammar analysis process to convert chemical nomenclature to CTs was by Elliot in 1969 [103]. Before this Garfield produced a system to calculate a compound's molecular formula from its name [104]. This algorithm did not include a complete grammatical description of chemical nomenclature, but all the basic facets of such a grammar were examined. Kirby *et al.* have been active in this area since 1985, when they presented a program that could convert an IUPAC systematic name to a chemical structure [105, 106, 107, 108, 109, 110]. The approach taken by this group was to create a formal grammar from the informal IUPAC rules then subsequently to modify a Simple Left Right parser generator (SLR) to apply to the context-free grammar. (A full discussion of how such a parser works and can be created can be found in section 3.1). The program has since been extended to find, and automatically correct, errors found in chemical names and to parse some semi-systematic names. However, in their own words, the work only focused on

> certain classes of compounds of industrial importance, including some cases of semi-systematic and trivial nomenclature.

The areas considered were, perhaps understandably, those that are the most tractable. The work covered much of the hydrocarbon nomenclature and has since been extended to recognise many acids, alcohols, aldehydes, ketones and ethers.

Ultimately, a program is envisaged that can identify (in a paper or thesis) a chemical name, a synthetic route and the analytical data for this compound. A CT would be created for the title compound, and all chemicals identified in the synthesis. Automated validation could be achieved by generating various chemical properties, using either *ab initio* or semi-empirical methods. These would be checked against the reported data and any anomalies reported, this operation could also remove any ambiguities present in the structure generated. Calls to an outside program for reaction-prediction could also be made to verify that the synthetic route reported did, in fact, lead to the compound reported. It is hoped that such a program will prevent problems such as that described in section 1.3.

The current methodology approaches the problem from both ends. Lexemes are currently being developing to tokenise a chemical name. During the period of this author's involvement with the project, only sections A to C of the IUPAC blue book [111] were considered but eventually the entire book will be encoded. Methods are also being created to store, join and otherwise manipulate molecular fragments. Methods to deal with brackets were developed, allowing the program to know which bracket level it is processing. Although a formal grammar to describe the syntax of chemical names must be fully developed, collaboration with Natural Language Processing groups have led to the consideration of including machine learning routines in conjunction with this. With such an implementation it would no longer be necessary to manually update and improve the lexemes, the grammar or the library of known fragments.

A combination of lexical and syntactic analysis should be able to process the characters in the chemical name

1,2–dichlorohexane

into the following tokens:

1. The *locant* `1`
2. *Comma*
3. The *locant* `2`
4. *Hyphen*
5. The *multiplier* `di`
6. The *halogen* `chlor`
7. *omark*
8. The *chain* `hex`
9. The *saturation* `ane`

The blanks separating the characters of these tokens would normally be eliminated during preprocessing. Recursive hierarchical syntactic analysis allows the construction of the parse tree (see section 3.1) shown in figure 2.20. From this, the CT of the molecule is immediately recoverable.

Chemists often prefer to have the ability to perform graphical as well as text searching for complex chemical structures and sub-structures mentioned in scientific literature. To date this has only been possible if the structure has been rendered searchable by the inclusion of the appropriate CT or SMILES [112, 113, 114] string to represent the molecule. If a compound, chemical name or component has only been mentioned in the document as a text string it has not been possible to search for these using a graphical search engine.

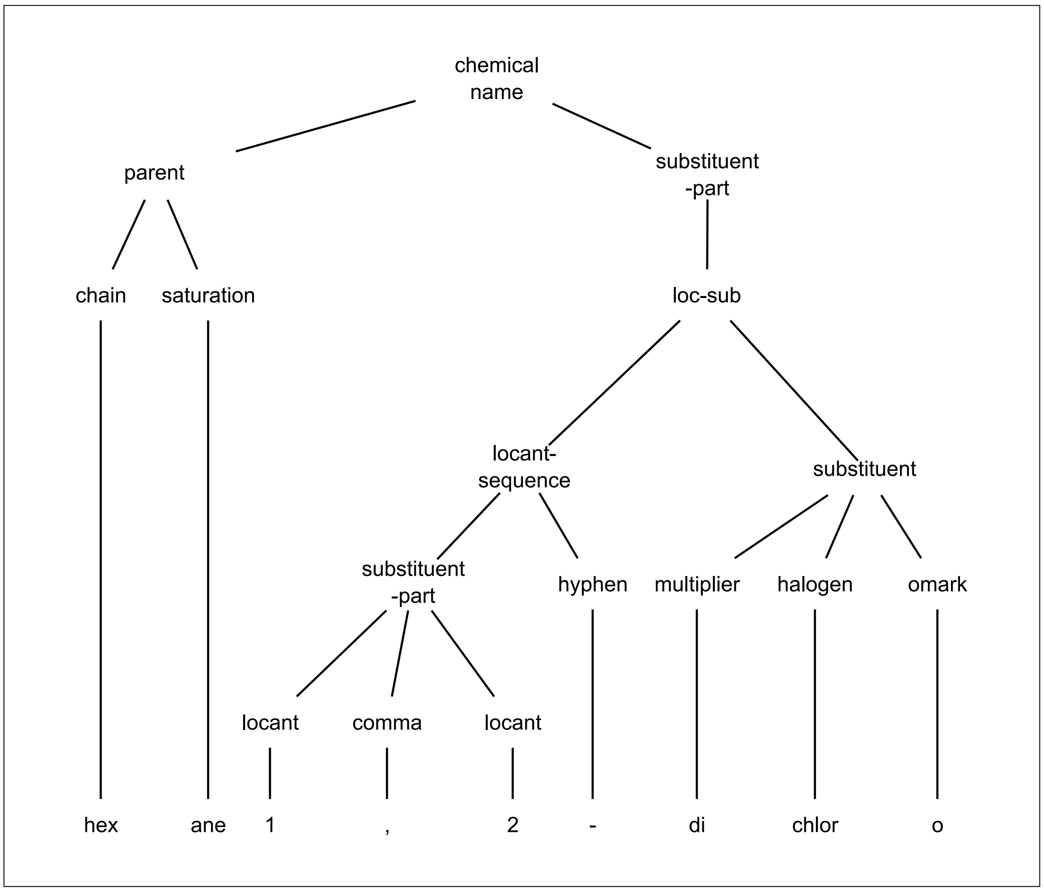


Figure 2.20: The parse tree of the chemical name 1,2–dichlorohexane.

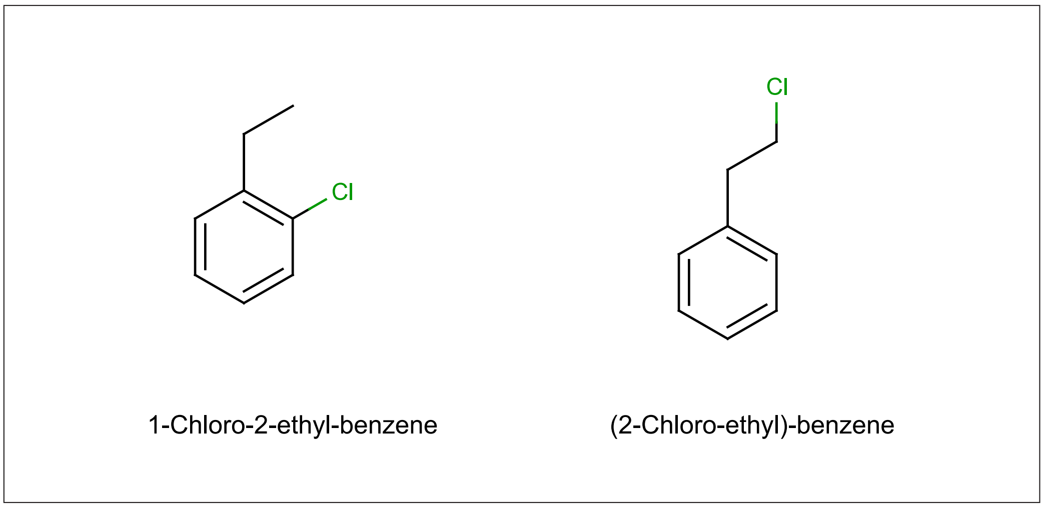


Figure 2.21: Two structures that might be referred to as 2-Chloroethyl benzene

The SciBorg project [115] has focused on improving and implementing much of the functionality mentioned above with Corbett, Copestake *et al.* demonstrating proof of concept, or better, implementations of much of the technology [116, 117, 118]. This has included taking a non-deterministic approach toward chemical name parsing. 2-Chloroethyl benzene is an ambiguous name because insufficient locants have been specified. The name might be used to describe both the compounds in figure 2.21 (which also shows a grammatically-correct name for each [119]); both structures can be reconstructed using non-deterministic approaches.

2.6 Conclusions

The work above shows that it is possible for a machine to read and extract data from the highly-structured analytical data section in the legacy formats currently used to publish organic chemistry. The extraction can be performed with very high recall and precision rates using (hierarchical) regular expressions although the process was more difficult than expected. However, whilst the recall and precision rates are high (and are being improved by implementing new techniques) they are not currently sufficiently high to allow the full automation of the process. Parsing the synthetic methodology also proved to be far less tractable than expected.

The work has also shown that chemical names cannot be identified reliably solely by using regular expressions. However, other methods do allow high rates of both recall and precision. The lack of a CT in machine-understandable form and the inability to reliably produce the correct CT from the information available has been highlighted as a major problem which must be addressed — for molecule-based data-driven science to be possible, it is vital that a CT is available.

Supplementary data provided with articles may include the input files and results of computational chemistry calculations or the crystal structures determined. These data formats are more structured and, importantly, should necessarily contain the CT for the molecule (or molecules) in an almost trivially recoverable form. Data in such forms is considered in the subsequent chapters.

Chapter 3

Parsing Program Input — Compilers

The previous chapter dealt with the extraction of data from the semi-structured experimental section of chemical papers which did not conform to a formal grammar. It should however be possible to create a formal grammar for data produced by, or for, a computer program. Data in such a form should therefore lend itself to automated extraction with extremely high rates of both recall and precision; thus eliminating, or at the least drastically reducing, the requirement for human intervention.

A compiler is a program that reads a program in one language — the *source* language — and translates it into another language — the *target* language (figure 3.1). Both the source and target languages should have fully specified grammars; part of the translation process involves the compiler reporting any errors in the source program (deviations from the specified grammar).

To avoid having to use proprietary software, only ASCII files are considered as suitable for use as the source language. In all cases the eventual target language is CML, although in some cases XML is used as the primary target language which is then transformed to CML using stylesheets. There follows a general introduction to compiler theory and how it has been applied to extract data from input files for computational chemistry programs [120].

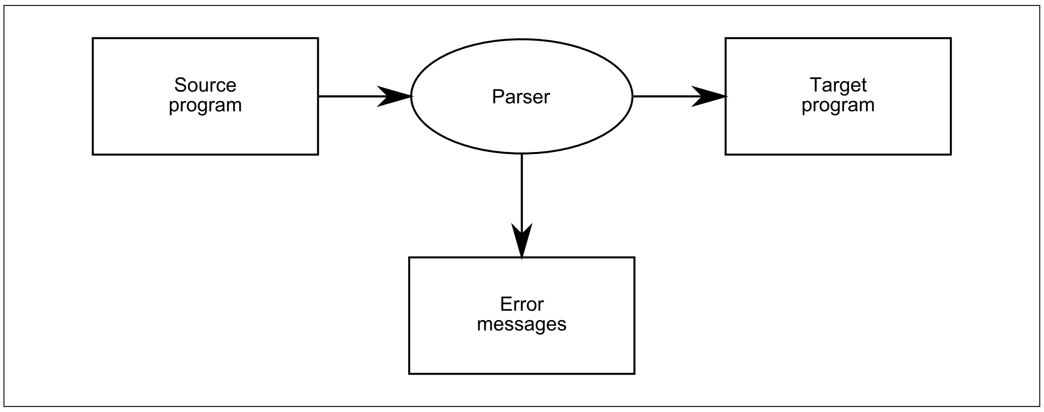


Figure 3.1: Overview of a compiler.

3.1 Compilers and Classification Techniques

There are two parts to the compilation process: analysis and synthesis. The analysis breaks the source program up into constituent pieces and may create an intermediate representation of the data. The synthesis part constructs the desired target program from the constituent pieces of the intermediate representation. This is often the most complex part for code compilation. In contrast, when dealing with chemistry held in an XML form, the synthesis is almost trivial and involves the application of stylesheets. During analysis the operations implied by the source program are determined and recorded in a hierarchical structure called a tree. A specialised kind of tree called a syntax tree is often used, in which each node represents an operation and the children of the node represent the arguments of the operation. Syntax trees are not required for processing computational chemistry but are vital for less structured chemical data.

A compiler itself is often not sufficient to create an entire target program because the source program may be stored in separate files, or have *include* statements for brevity. A pre-processor usually deals with the task of pulling together all the relevant pieces of the source program. It was necessary to implement a pre-processor in some cases, although it was decided that *include* statements would not be expanded.

3.1.1 The Phases of a Compiler

The traditional view of the phases of a compiler and the way in which they interact is shown in figure 3.2. This project only required five of these phases: *lexical analysis*, *syntactic analysis*, *semantic analysis*, *symbol-table manager* and the *error handler.* A symbol-table is a data structure containing a record for each identifier with fields for the attributes of the identifier, allowing the rapid recall and modification of data relevant to that record. These tasks are managed by using a XML infrastructure.

Every phase may encounter or generate errors. All errors must be dealt with, so that the compilation (transformation) process can proceed, allowing further errors to be identified although some particular errors would immediately stop the process. The phases which most frequently give rise to the largest fraction of the errors are syntax and semantic analysis. The lexical phase can detect errors where the characters remaining in the input do not form any token of the language.

The analysis of the source program consists of three phases:

1. *Linear analysis*, in which the stream of characters making up the source program is read from left-to-right and grouped into tokens that are sequences of characters having a collective meaning.
2. *Hierarchical analysis*, in which characters or tokens are grouped hierarchically into nested collections with collective meaning.
3. *Semantic analysis*, in which certain checks are performed to ensure that the components of the program fit together meaningfully.

In a compiler, linear analysis is called *lexical analysis* or scanning. For the java assignment

```
int position = initial + rate * 60;
```

the lexical analyser would produce the stream of tokens, where each token represents a logically cohesive set of characters, such as identifier, keyword

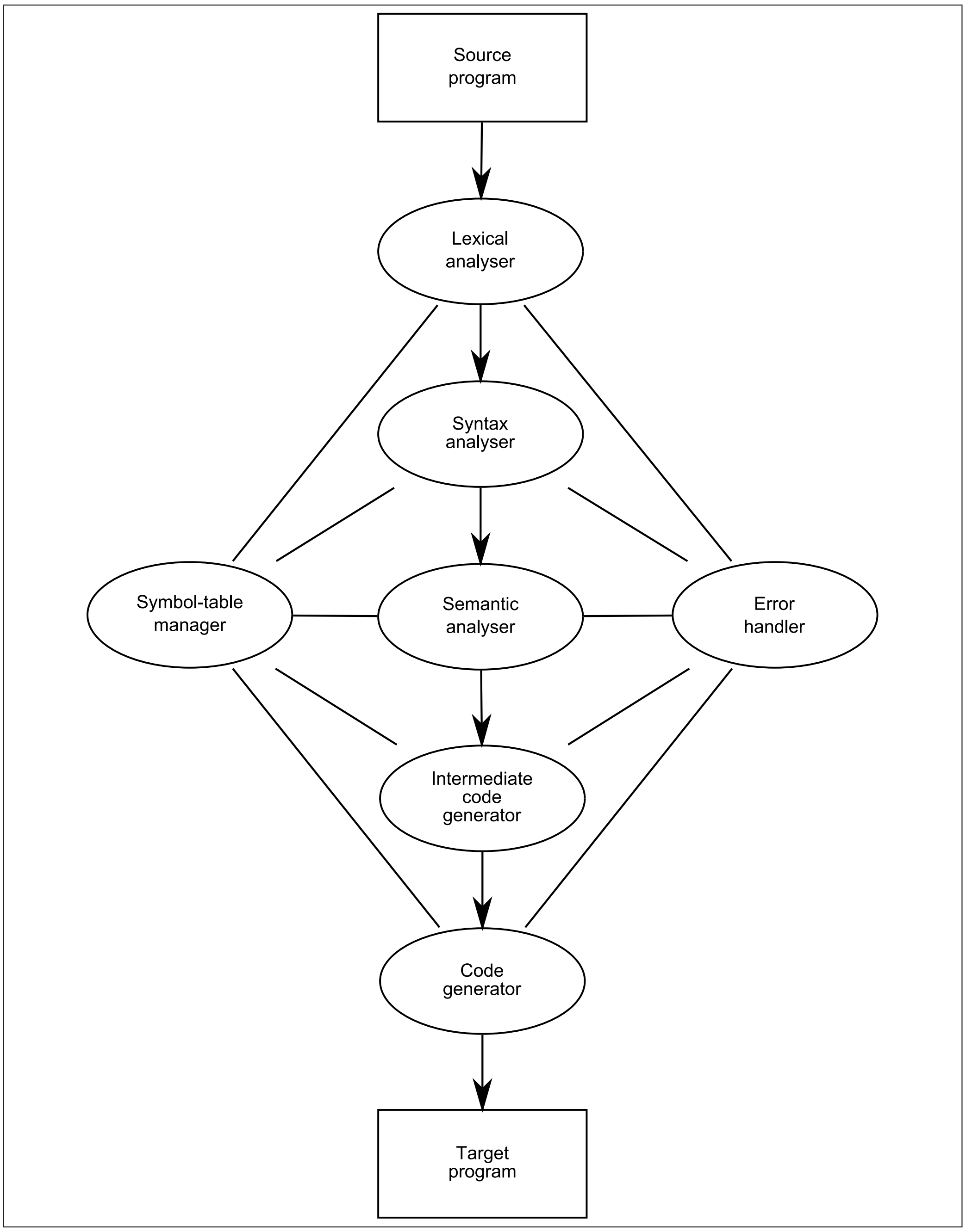


Figure 3.2: The phases of a compiler.

(for example `if`, `while` and `int`), or a punctuation character. The character sequence forming a token is called the *lexeme* for the token. In this example `id1`, `id2` and `id3` will be used to represent position, initial and rate respectively, emphasising that the internal representation of an identifier is different from the lexeme for that identifier. Figure 3.3 shows how this assignment would be processed by the three phases. The semantic analyser allows the compiler to know that an integer must be formed by the addition or multiplication of two integers. This allows the assignment of `id2`, `id3` and `60` to type `int`.

The division between lexical and syntactic parsing is chosen to simplify the overall analysis task, although the division becomes necessary if the source language is inherently recursive. Lexical constructs do not require recursion, while syntactic constructs often do. Context-free grammars are a formalisation of the recursive rules that can be used to guide syntactic analysis (discussed later in this chapter). For example, recursion is not necessary to recognise a number, but is required to match parentheses in expressions. For code generation, the semantic analysis phase checks the source program for syntactic errors and gathers type information for the subsequent code generation phase.

The Number of Passes

In the compilation of computer code, several phases are usually implemented in a single pass: where a pass is defined as reading an input file and writing an output file. It is desirable to keep the number of passes to a minimum, but whilst this used to be necessity it is now more of a guideline. The lack of definite reserved-words in chemical literature means that a one-pass compiler is impossible.

A Simple Compiler

The syntax of a language can be represented using a notation called context-free grammars or Backus-Naur Form (BNF) [121, 122]; the specification is shown in appendix C. BNF came about as part of the creation process for

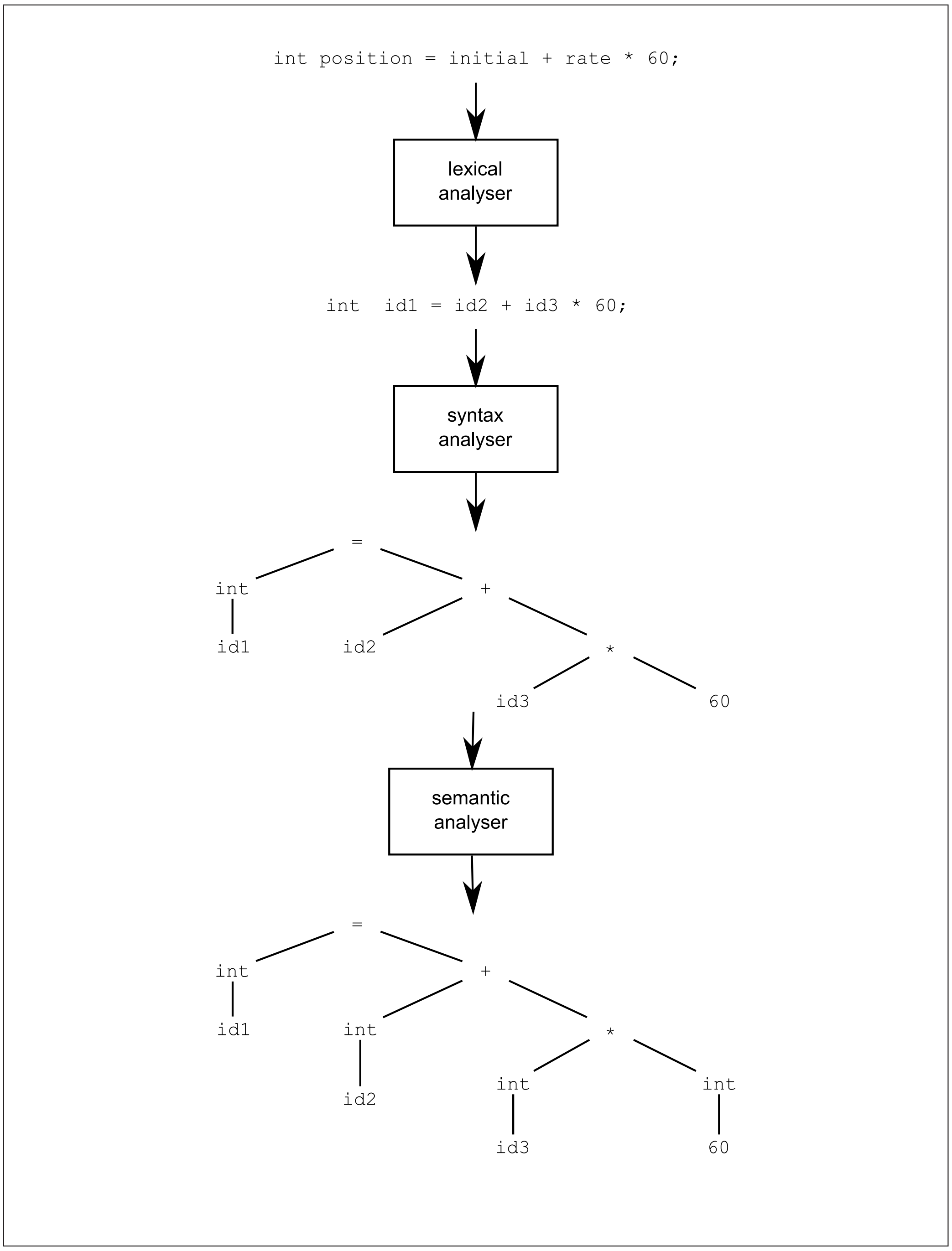


Figure 3.3: Assignments produced by the first three phases of a compiler.

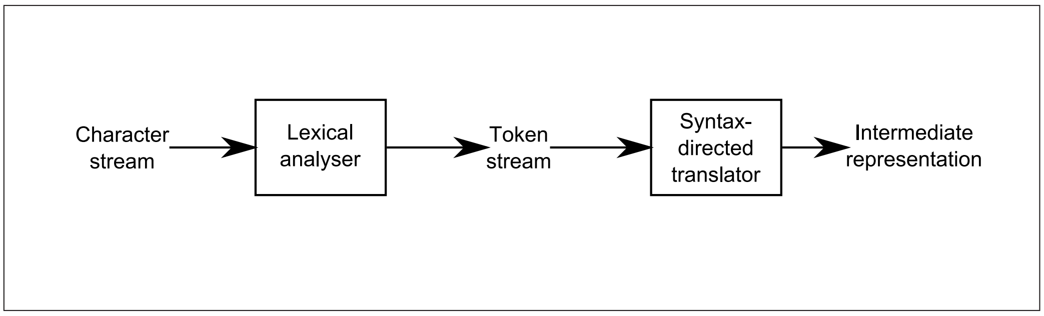


Figure 3.4: The structure of a compiler incorporating a syntax-directed translator.

ALGOL. At the first World Computer Congress, which took place in Paris in 1959, Backus presented a formal description of the international algebraic language which was later called ALGOL 58 [123]. The formal language he presented, which would later evolve in to the BNF, was based on Post's production system [124].

When such a context-free grammar exists it may be used to guide the translation of a program; this is called syntax-directed translation. The structure for such a compiler is shown in figure 3.4.

3.1.2 Context-Free Grammars

A context-free grammar (grammar for short) is a way of specifying the syntax of a language (or any data). This is most easily illustrated with reference to a programming language. An `if-else` statement in Java has the form:

```
if ( expression ) statement else statement
```

In other words, the statement is a concatenation of the keyword `if`, an opening parenthesis, an expression, a closing parenthesis, a statement, the keyword `else`, and another statement. Using the variables `expr` and `stmt` to represent an expression and a statement respectively this may now be written

```
stmt ⇒ if ( expr ) stmt else stmt
```

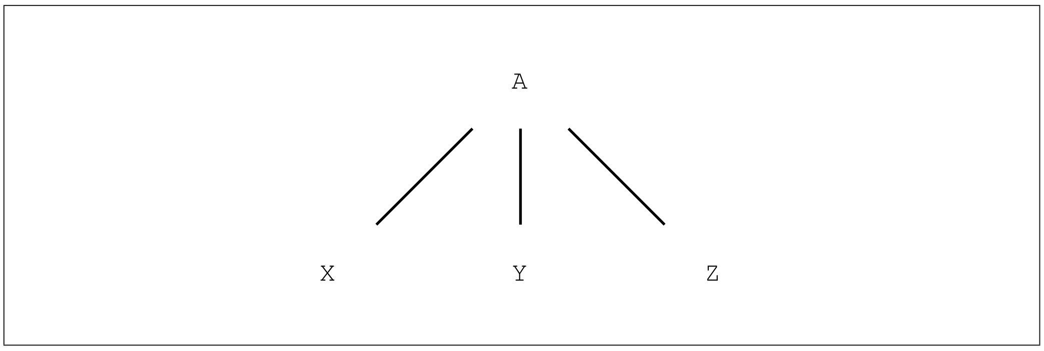


Figure 3.5: A simple parse tree.

where the arrow should be read ‘may have the form’. Rules of this form are termed productions. In productions, lexical elements such as the keywords and the parentheses are called tokens, whilst variables such as `expr` and `stmt` comprise collections of tokens and are called nonterminals.

A context-free grammar has four components:

1. A set of tokens, known as terminal symbols.
2. A set of nonterminals.
3. A set of productions where each production consists of a nonterminal, called the left side of the production, an arrow, and a sequence of tokens and/or nonterminals, called the right side of the production.
4. A designation of one of the nonterminals as the start symbol.

3.1.3 Parse Trees

A parse tree pictorially shows how the start symbol of a grammar derives a string in the language. If nonterminal A has a production

$$A \Rightarrow XYZ$$

then a parse tree may have an interior node labelled A with three children X, Y and Z from left to right (figure 3.5). Formally, given a context-free grammar, a parse tree is a tree with the following properties:

- the root is labelled by the start symbol
- each leaf is labelled by a token or by ϵ
- each interior node is labelled by a nonterminal
- if A is the label of an interior node and X_1, X_2, …, X_n are the labels of the children of that node from left to right, then A $\Rightarrow$ $\mathrm{X}_1\mathrm{X}_2\ldots\mathrm{X}_n$ is a production. Here X_1, X_2, …, X_n stand for a symbol that is either a terminal or a nonterminal. If $A \Rightarrow \epsilon$ then each node labelled A may only have a single child labelled ϵ

A simple example of a parse tree was seen in figure 3.5. The leaves of a parse tree read from left to right form the *yield* of the tree. Most parsing methods fall in to one of two classes, called *top-down* and *bottom-up* methods. These terms refer to the order in which the nodes in the parse tree are constructed. In the former, construction starts at the root and proceeds toward the leaves and vice versa for the latter. Top-down parsers are usually easier to construct by hand and this is the general approach taken for this project.

Ambiguity

A grammar is said to be *ambiguous* if it is possible to represent a particular token string by more than one parse tree. Since this would generally mean that the token string's meaning can be interpreted in more than one way it is usual to attempt to remove all ambiguity in a grammar. It is possible to use ambiguous grammars, but these require additional rules to resolve the ambiguities.

3.1.4 Predictive Parsers

Recursive-descent parsing is a top-down method of syntax analysis in which a set of recursive procedures are executed to process the input. Predictive parsing is a special case of this where the *lookahead symbol* unambiguously determines the procedure selected for each nonterminal. The lookahead symbol is the next token in the token stream. A predictive parser consists of a procedure for every nonterminal:

1. Each procedure decides which production to use by looking at the lookahead symbol; the production, with right side α, is used if lookahead symbol is FIRST(α). If there is a conflict between two right hand sides for any lookahead symbol then the method cannot be used. A production with ϵ on the right hand side is used if the lookahead symbol is not the FIRST set for any other right hand side.

2. The procedure uses a production by mimicking the right hand side. A nonterminal calls the procedure for that nonterminal, and a token matching the lookahead symbol results in the next input token being read. If at some point the token in the production does not match the lookahead symbol, an error is declared.

FIRST is defined such that, if a nonterminal α that has the production $\alpha \Rightarrow \beta\chi\delta\ldots\zeta$ then FIRST(α) is β. It is impossible to construct a predictive parser for chemical documents owing to the number of nonterminals that share the same FIRST symbol (see section 3.3).

3.1.5 Lexical analysis

A lexical analyser reads and converts the input into a stream of tokens to be analysed by the parser. From the definition of a grammar: a sentence of a language consists of strings of tokens. A sequence of input characters that comprises a single token is called a *lexeme*. Examples of possible lexemes for chemical names are seen later in this chapter. A lexical analyser insulates a parser from the lexeme representation of tokens. For instance, instead of passing the actual value of a number in the source to the parser, the lexical analyser would identify the number and pass a token indicating that a number was present to the parser, with a pointer to the actual value (see figure 3.6). It may also remove or normalise whitespace and comments.

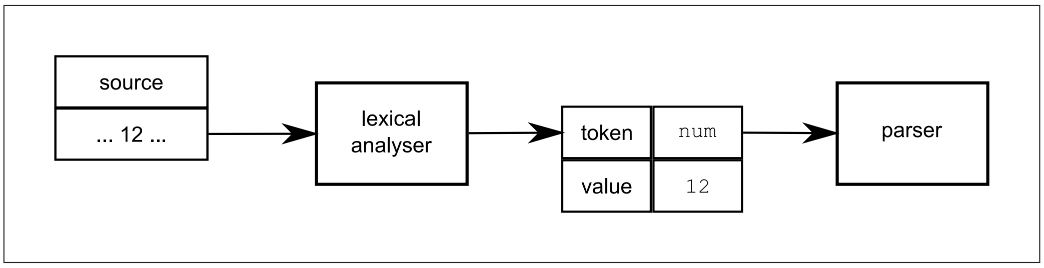


Figure 3.6: An example of how a lexical analyser passes tokens to the parser.

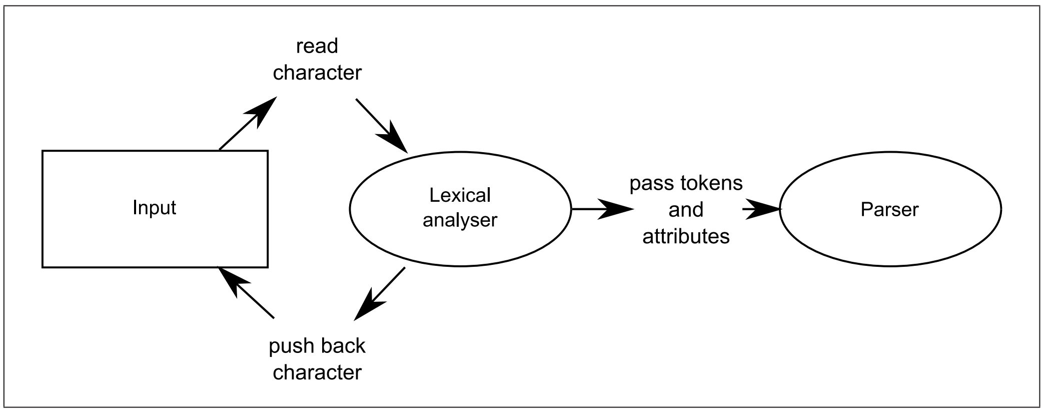


Figure 3.7: The lexical analyser and parser acting as a producer-consumer pair.

The interaction of a lexical analyser, the input and the parser are shown in figure 3.7. The analyser reads characters from the input and determines what token will represent them. In some cases it is necessary to lookahead several characters to determine what the current token should be. Once these characters have been read they must be *pushed back* on to the input in case they form the start of a new token. The lexical analyser and parser form a *producer-consumer* pair. The analyser produces tokens and the parser consumes them.

3.1.6 Regular Expressions

Lexemes are usually determined by *matching* against a regular expression. Although regular expressions have previously been discussed informally, a more rigorous definition follows.

A regular expression is built up from simpler regular expressions using a set of defining rules. Each regular expression r denotes a language $L(r)$. The defining rules specify how $L(r)$ is formed by combining, in various ways, the languages denoted by the sub-expressions of r. The following rules show the definition of the languages denoted by the regular expression being defined:

1. ϵ is a regular expression that denotes ϵ, that is, the set containing the empty string.

2. If a symbol a is a symbol in Σ, then a is a regular expression that denotes a, i.e., the set containing the string a. Although the same notation is used for all three, technically the regular expression a is different from the string a and the symbol a. It should be clear from the context whether a is being treated as a regular expression, string or symbol.

3. Suppose r and s are regular expressions denoting the languages $L(r)$ and $L(s)$. Then:

   (a) r|s is a regular expression denoting $L(r) \bigcup L(s)$.

   (b) rs is a regular expression denoting $L(r)L(s)$.

   (c) r* is a regular expression denoting $(L(r))^*$.

   (d) (r) is a regular expression denoting $L(r)$. This rule states that extra parentheses may be placed around regular expressions if desired.

A language denoted by a regular expression is said to be a *regular set.* The specification of regular expression is an example of a recursive definition. Rules (1) and (2) form the basis of the definition and rule (3) provides the inductive step. To avoid unnecessary parentheses in regular expressions, the following conventions have been adopted.

1. The unary operator * has the highest precedence

2. Concatenation has the second highest precedence

3. The or operator | has the lowest priority

All the operators are left associative. Under these conventions, (a)|((b)*(c)) is equivalent to a|b*c. Both expressions denote the set of strings that are either a single *a* or zero or more *b*'s followed by one *c*.

Some languages cannot be described by any regular expression. One example is the set of all strings of balanced parentheses (in fact regular expressions cannot be used to describe any arbitrary set of balanced or nested constructs but a context-free grammar can). Regular expressions can only be used to denote a fixed or arbitrary number of repetitions of a given construct. Thus Hollerith strings of the form n`H`$\mathtt{a}_1\mathtt{a}_2$ `...` $\mathtt{a}_n$ from early versions of Fortran cannot be described, because the number of characters following `H` must match the decimal number n before H. Matching strings of this type is possible using a library such as Perl Compatible Regular Expressions which allow both look ahead and recursion functionality. However such regular expressions no longer correspond to the original mathematical definition of regular expressions [125].

3.1.7 Non-deterministic parsing

Context-free grammars and regular expressions are deterministic methods of identifying data, *i.e.* the behaviour of the system is described completely without probabilities (other than zero or one). Non-deterministic methods (such as Naïve Bayesian), which are largely used for classification, are discussed below.

Hidden Markov Model

Hidden Markov Models (HMMs) are often used in biochemistry to predict which amino acid residue is most likely to occur next in a chain [126, 127]. The technique uses a training set that has been manually marked up to determine initial transition probabilities, which become modified as the program sees more examples.

Formally, the HMM is a finite set of states, each of which is associated with a (generally multidimensional) probability distribution. Transitions between states are governed by a set of probabilities called transition probabilities. In a particular state an outcome or observation can be generated, according to the associated probability distribution. Only the outcome, not the state, is visible to an external observer; hence the name Hidden Markov Model.

The theory is based on three assumptions:

1. That the next state is dependent only upon the current state, and the resulting model becomes actually a first order HMM. However, generally the next state may depend on past k states and it is possible to obtain such a model, called an kth order HMM. A higher order HMM will have a higher complexity. Even though the first order HMMs are the most common, some attempts have been made to use the higher order HMMs (the Markov assumption).

2. That state transition probabilities are independent of the actual time at which the transitions takes place (the stationary assumption).

3. That the current output (observation) is statistically independent of the previous outputs (observations). Unlike the other two, this assumption has a very limited validity. In some cases this assumption may not be fair enough and therefore becomes a severe weakness of the HMMs.

3.1.8 Bayesian Classification

In the Bayesian approach to statistical inference, probability is a model of scientific knowledge [128]. One of the strengths of the Bayesian method is that it allows expert knowledge, in the form of a prior probability distribution, to be formally incorporated into the statistical analysis. The Bayesian paradigm views both the data and the underlying parameters that generated the data as random variables — random because they are unknown [129].

A Bayesian classifier is commonly employed in email filters to separate spam from non-spam. The expert knowledge is incorporated by producing a list of the features that are, in the experts' opinion, most likely to indicate that an email is spam. The list is likely to contain words such as *Viagra* and *porn*. These would be described as information-rich words because their presence is highly indicative that the email can be classified as spam. Information-poor words might be *dear* or *cost* which are likely to appear with approximately equal frequency in emails of both type. Once the initial list of features has been created it may be augmented by the classification program if it identifies other information-rich words.

Natural Language Processing

The goal of Natural Language Processing (NLP) is to allow machines to analyse, understand and generate languages that humans use naturally [130]. This is a hard task. Human language contains much ambiguity and sentence construction often makes comprehension difficult. The sentences below are both human-understandable but present problems to machines:

> I can can a can.
>
> A well-dressed man was speaking; he had a foreign accent.

The first sentence contains the word *can* present in three different forms: a modal modifier of a verb, as a verb and as a noun. Syntax analysis would group the first two occurrences as a modal verb. The second sentence contains anaphora; this is coreference of one expression with its antecedent. Anaphora is common because it avoids repetition of words, making sentences more elegant. Part-of-speech analysis and tagging allows many of the common linguistic problems to be resolved.

Shallow parsing is the process of identifying syntactic phrases (such as noun phrases) in natural language sentences, for instance, Chomsky's system of transformational grammar. As outlined in Syntactic Structures [131], it comprised three sections, or components; the phrase-structure component,

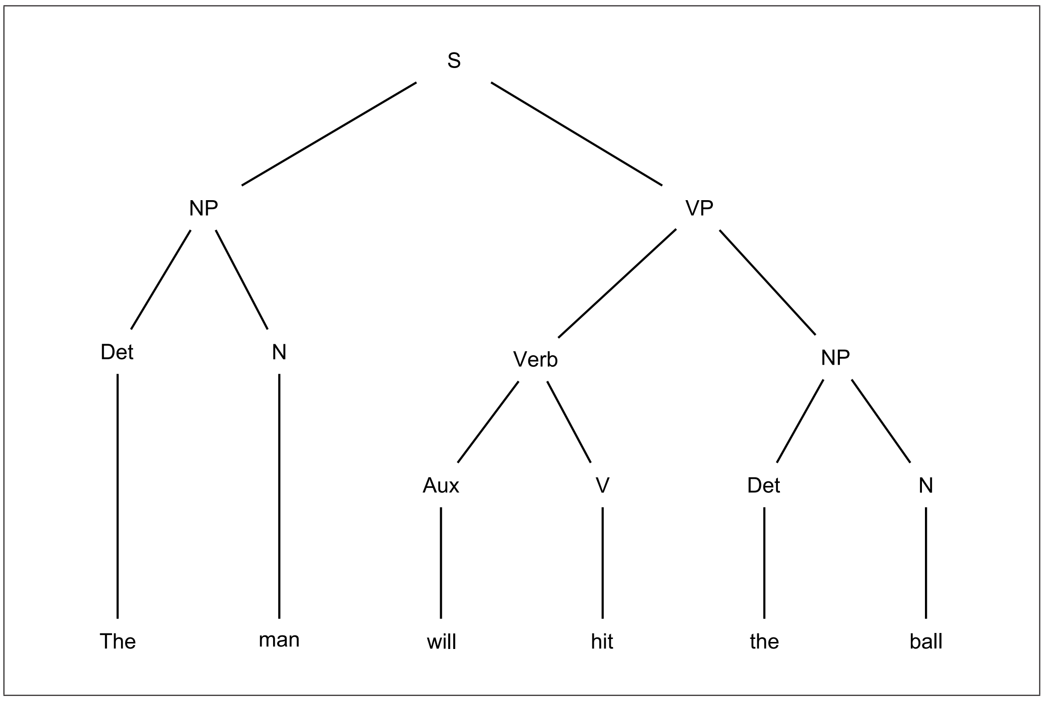


Figure 3.8: The parse tree generated by the Chomskian analysis of the phrase 'The man will hit the ball'.

the transformational component, and the morphophonemic component. In the following system of rules, `S` stands for Sentence, `NP` for Noun Phrase, `VP` for Verb Phrase, `Det` for Determiner, `Aux` for Auxiliary (verb), `N` for Noun, and `V` for Verb stem.

1. `S` → `NP` + `VP`
2. `VP` → `Verb` + `NP`
3. `NP` → `Det` + `N`
4. `Verb` → `Aux` + `V`
5. `Det` → the, a, . . .
6. `N` → man, ball, . . .
7. `Aux` → will, can, . . .
8. `V` → hit, see, . . .

This is a simple phrase-structure grammar. It generates and thereby defines as grammatical many sentences and it assigns to each sentence that it generates a structural description. Figure 3.8 shows the parse tree generated by Chomskian analysis of the phrase ‘The man will hit the ball’.

3.2 JFlex and CUP

Several tools exist to create lexical analysers from special purpose notation based on regular expressions. This project used JFlex [132], which is a Java implementation of the *C* program *flex* [133, 134]. Figure 3.9 shows a typical example of this process. A specification of a lexical analyser is created in the JFlex language, this is run through the JFlex compiler to create the Java code. This code consists of a tabular representation of a transition diagram constructed from the regular expressions specified, together with a standard routine to recognise lexemes. The Java code is then compiled using the `javac` command. The resultant class is the lexical analyser that transforms an input stream into a sequence of tokens.

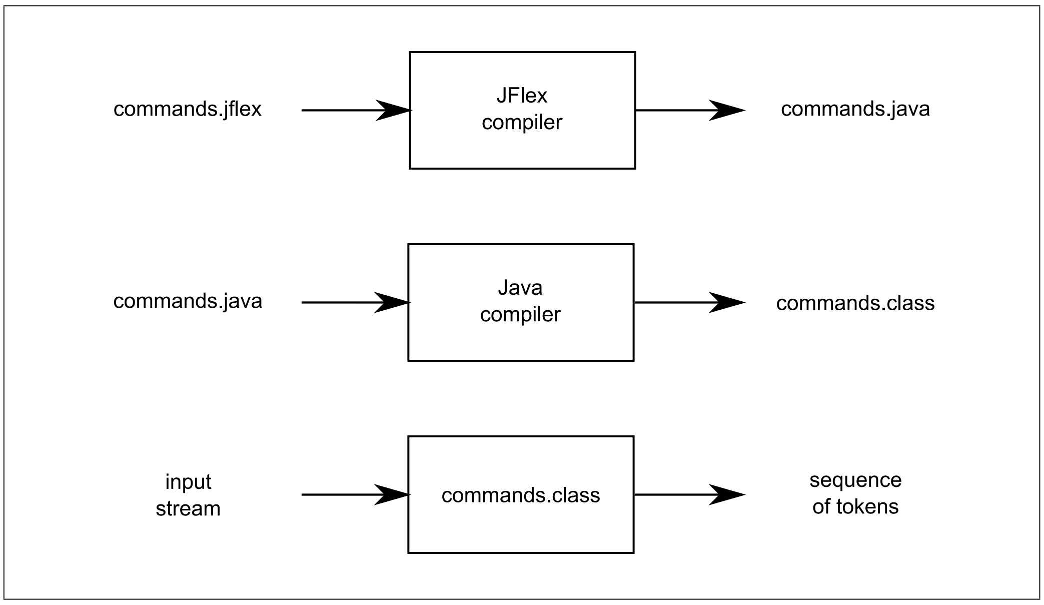


Figure 3.9: Creating a lexical analyser with JFlex.

```
USER CODE
- Comments and import statements
- Anything in here is placed verbatim in Lex.java
%%
OPTIONS AND DECLARATIONS
- Directives and macros
%%
LEXICAL RULES
- Regular expressions and Java actions
- Rules section
```

Figure 3.10: The specifications of a JFlex file.

```
%%
%class CommentRemover
%standalone
%unicode
LineTerminator = \r|\n|\r\n
Comment = ";".*{LineTerminator}
AnythingElse = [^;]+
%%
{Comment}                { ; /*don't print these out */ }
{LineTerminator}         { System.out.println(); }
{AnythingElse}           { System.out.println(yytext()); }
<<EOF>>                  { return 0; }
```

Figure 3.11: The comment remover pre-processor in JFlex. The token `<<EOF>>` is pre-defined and represents the end of the file. `yytext()` commands the program to replace this phrase with whatever was matched by the token.

Figure 3.10 shows the JFlex file specification (the BNF of the lexical rules is shown in appendix D). If the `%standalone` directive is present the Java code in the lexical rules usually directly creates the target language otherwise it returns a token for the parser generator to consume. An example of a standalone scanner is shown in figure 3.11, where the target language is simply the original file with the comments removed.

When consuming its input, the scanner determines the regular expression that matches the longest portion of the input (longest match rule). If there is more than one regular expression that matches the longest portion of input (*i.e.* they all match the same input), the generated scanner chooses the expression that appears first in the specification. After determining the active regular expression, the associated action is executed. If there is no matching regular expression, the scanner terminates the program with an error message.

The scanner is used to determine that only permissible lexemes are present in the input. However the meaning of each lexeme (and how it should be processed) is not always apparent without context; for instance, a token representing a number might represent the atomic mass, a coordinate, the year or the amount of memory used. Therefore a syntactic analyser must be introduced. A Look-Ahead Leftmost Reduction (LALR) parser is a specialised

```
PairBlock ::= PAIR:t1 BLANKLINES PairLineBlock
            ;

PairLineBlock ::= PairLine
                | PairLineBlock PairLine
                ;

PairLine ::= PairLine1
           | LongPairLine1
           | PairLine2
           ;

PairLine1 ::= SPACE INT:i1 SPACE INT:i2 SPACE INT:i3 SPACE FLOAT:f1 SPACE
FLOAT:f2 EOL

{: System.out.println("<pair atom1='"+i1+"' atom2='"+i2+"' function='"+i3+"'
param1='"+f1+'"' param2='"+f2+"' />");
:}
   ;

LongPairLine1 ::= SPACE INT:i1 SPACE INT:i2 SPACE INT:i3 SPACE FLOAT:f1
SPACE FLOAT:f2 SPACE FLOAT:f3 SPACE FLOAT:f4 EOL

{: System.out.println("<longpair atom1='"+i1+"' atom2='"+i2+"'
```

Figure 3.12: The productions for a [`pair`] block.

```
terminal SPACE, EOL;

terminal String INT, PAIR, FLOAT;

non terminal PairBlock, PairLineBlock, PairLine, PairLine1, LongPairLine1, PairLine2;
```

Figure 3.13: Definitions of terminal and non terminal tokens.

form of a LR parse that can deal with more context-free grammars than Simple LR (SLR) parsers but fewer than LR(1) parsers. Constructor of Useful Parsers (CUP) is a system for generating LALR parsers from simple specifications [135]. CUP is a Java implementation of the widely used program YACC and is used to specify a grammar in BNF and how the various tokens should be processed [136]. Figures 3.12 and 3.13 show typical terminal and non-terminal token definitions and how they are combined to create a sample grammar.

3.3 Parsing Program Input

The input file for a computational chemistry program must be machine readable and machine parsable (hence highly structured and conforming to a

```
#include "ffgmx.itp"
#include "spc.itp"

 [ moleculetype ]
 ;name nrexcl
DRG      5
 [ atoms ]
 ;  nr  type resnr resid  atom  cgnr charge
     1     O     1 DRG      O8     1    0.000
  46
     8     H     1 DRG     HAB     1    0.280
 [ bonds ]
 ;ai  aj  fu    c0          c1
     1   2   1 0.123    502080.0 0.123    502080.0 ;    O8   C3
   46
     7   8   1 0.100    374468.0 0.100    374468.0 ;    N4  HAB
 [ pairs ]
 ;ai  aj  fu    c0          c1
     1   4   1                  ;    O8   C7
   46
     5   8   1                  ;    HA   N6
 [ angles ]
 ;ai  aj  ak  fu    c0          c1
     1   2   3   1 120.0       418.4 120.0       418.4 ;    O8   C3   N2
   46
     4   7   8   1 120.0       418.4 120.0       418.4 ;    C5   N4  HAB
 [ dihedrals ]
 ;ai  aj  ak  al  fu    c0    c1 m  c0   c1 m
     2   5   3   1   2   0.0 1673.6 0   0.0 1673.6 0 ; IDI    C3   N4   N2   O8
   46
     7   6   4   8   2   0.0 1673.6 0   0.0 1673.6 0 ; IDI    N6   C5   N4   C3
 [ system ]
PRODRG in water
 [ molecules ]
 DRG   2
 SOL   2747
```

Figure 3.14: A GROMACS topology file.

grammar). Any deviation from the rules governing input format can be interpreted as errors.

GROMACS [137] (Groningen Machine for Chemical Simulations) is an Open Source engine to perform molecular dynamics simulations and energy minimisations and is often used to model the behaviour of small molecules in a solvent. The input file (topology file) is in ASCII format and well defined in the documentation; this presented an ideal opportunity to create and test a simple parser, which could be adapted to deal with more complex and less well defined source. The full definition of the GROMACS topology file can be found in appendix E, and an example of a topology file is shown in figure 3.14.

A comment in a topology file is surrounded by a semicolon and a newline character. Comments are included for human-readability and are ignored by GROMACS. A pre-processor was created to remove comments before passing the results to the parser. The pre-processor is a *lexer*, that is: it forms tokens from the input character stream and specifies what should be done when a particular token is produced. The lexer was written in JFlex, the complete lexer code is shown in figure 3.11.

The GROMACS topology file shown in figure 3.14 was passed through the lexer. The lines of the resultant file are ambiguous because, for example, a regular expression to match a line of numeric [`bonds`] data may also match a line of numeric [`pairs`] data. It is thus necessary to determine the syntax of the line before it can be parsed. Finite-state automata were investigated as a possible way to incorporate this functionality in the lexer. However, because GROMACS permits retrospective definitions (which may be present anywhere in the document) this approach was impossible.

The tokenisation and syntactic analysis processes were performed by JFlex and CUP respectively. Figure 3.12 shows the production for the data in the [`pair`] block and how the captured data should be held in an XML form. The terminal and nonterminal tokens are defined in figure 3.13.

This approach allowed correctly-structured GROMACS input files to be parsed — no attempt was made to parse incorrectly-structured files. An incorrectly-structured file may be parsable but the resulting output is likely to be nonsensical because the structure is used to determine meaning. If GROMACS is presented with an incorrectly-structured file it will not parse it, but will flag up an error.

3.3.1 Parsing Program Output

The parser-like approach worked for GROMACS input files. 100% of the correctly-structured files tested were correctly converted to CML and 100% of the incorrectly-structured files generated the appropriate error message. Although it is important to be able to parse input files, there is relatively little data gained when compared to that available in output files.

Program output is generated by a machine, and thus consists of a finite vocabulary. Typically this has a well-defined structure but error messages may appear at any point making perfect parsing almost impossible. The output is designed to be human-readable and understandable but not necessarily machine-understandable. There is a large quantity of data in this form (with the possibility of there being far more). If the data can be parsed into a machine-understandable form, it can relieve much of the more mundane analysis thereby making it much more attractive for re-use.

The success of the parser-like approach to extract data from input files led to the creation of an adapted version to interpret the less-structured MOPAC output (see figure 1.13). The parser-like approach was attractive because it supports the *or* operator (but not the *ampersand* operator). This would permit each section of the output to be combined with corresponding error messages connected by the *or* operator, thereby allowing a simple program construction with an extremely low failure rate. The subsequent parser worked well for toy documents, but an exponential combinatorial explosion of the number of states was encountered when applied to actual documents, which prevented the parser from completing the translation. Figure 3.15 shows the largest section of a document that the parser was capable of matching using this approach. The parser-like approach was abandoned as a method to parse the less-structured output documents and other methods examined. This culminated in the creation of JUMBOMarker which forms the basis of the following chapter.

```
*******************************************************************************
**                            MOPAC2002 (c) Fujitsu                          **
*******************************************************************************

                             PM3 CALCULATION RESULTS


*******************************************************************************
*                  MOPAC2002 Version 1.01     CALC.'D. Mon May 12 15:10:14 2003
*  PM3      - THE PM3 HAMILTONIAN TO BE USED
*
*
*
*                 CHARGE ON SYSTEM =     0
*
*
*
*  T=       - A TIME OF 86400.0 SECONDS REQUESTED
*  DUMP=N   - RESTART FILE WRITTEN EVERY  7200.000 SECONDS
*******************************************************************************
PM3 CHARGE=0
```

Figure 3.15: The largest section of the MOPAC output that was matched using JFlex.

Chapter 4

Parsing Program Output — JUMBOMarker

Current computational chemistry programs use input and output formats designed to be understood by a human; these files are usually ASCII text. Figure 4.1 shows various input and output schemes for programs and the various designs are discussed below.

Design 1 is the ideal case; XML/CML is used to hold all the data (both input and output). This allows all concepts to be fully marked up according to prearranged dictionaries. Such dictionaries are currently being created and tested by, amongst others, the Murray-Rust group [139]. Currently, this design is not usual practice, although if a program is Open Source, it may be achievable [140, 141, 142]. Holding both input and output data in XML/CML is advantageous because it allows multiple programs to be linked together simply (see figure 4.2). However since XML/CML is not *human-friendly*, stylesheets are used to convert to a preferred display format. If this were to be HTML-based (with SVG graphs incorporated) all the data could be easily hyperlinked to the appropriate dictionary. Unfortunately most computational chemistry programs are still written in Fortran which has little XML support, thus this design requires a lot of work.

Design 2 is possible if the source code is accessible, but is simpler than design 1 because it does not require XML support in the program code. Spe-

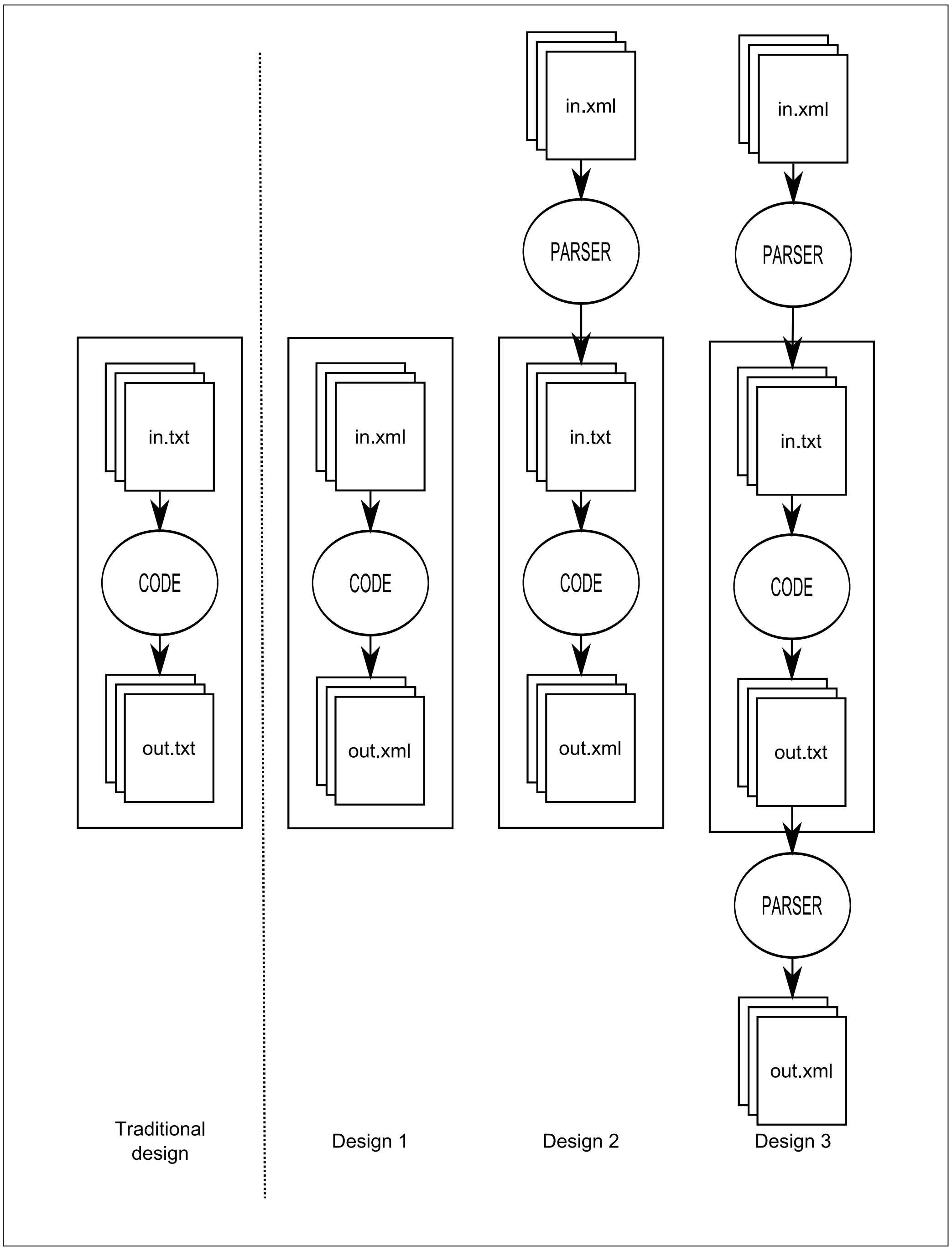


Figure 4.1: Program designs, with different input and output methods.

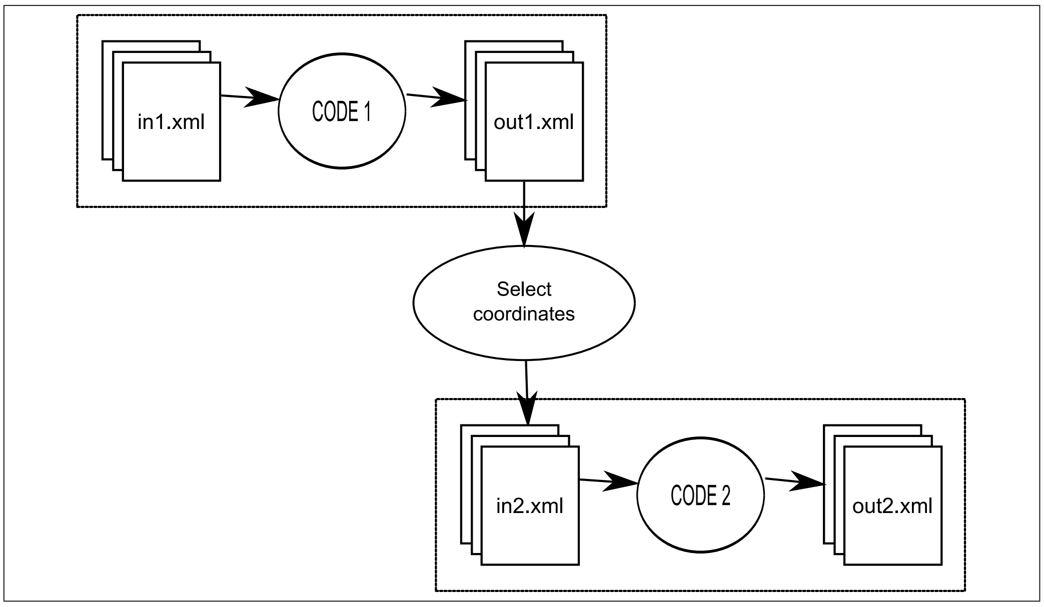


Figure 4.2: Programs can be linked together using XML/CML 'glue'.

cific parsers (usually stylesheets) must be constructed for each code requiring the creation of input. As XML is ASCII, the original *out* statements can be replaced with a new statement to output the data in XML format. The XML is thus never held as a object in memory, merely written out line by line to the output file.

Design 3 does not require access to the source code. Effectively the original program is wrapped by XML-aware parsers, the input parsers being identical to that in design 2. However the output parser must convert text to XML/CML. Whilst this approach will work for any code, it may require more user intervention (for instance moving the files between the program and the parsers). This design is more fragile that the others because any changes to input and output format of the code requires the creation of a new parser. The text to XML parser created for this process is JUMBOMarker.

JUMBOMarker provides the user with a way to convert computational chemistry output into XML and subsequently to convert this into CML or

```
<!ELEMENT template (moduleGroup | primitiveGroup)*>
<!ATTLIST template
    default       CDATA   #IMPLIED
>
<!ELEMENT moduleGroup (moduleGroup | primitiveGroup)*>
<!ATTLIST moduleGroup
    id            CDATA   #IMPLIED
    name          CDATA   #IMPLIED
    ref           CDATA   #IMPLIED
    maxOccurs     CDATA   #IMPLIED
    minOccurs     CDATA   #IMPLIED
    splitBefore   CDATA   #IMPLIED
>
<!ELEMENT primitiveGroup (primitive*)>
<!ATTLIST primitiveGroup
    id            CDATA   #IMPLIED
    name          CDATA   #IMPLIED
    ref           CDATA   #IMPLIED
    maxOccurs     CDATA   #IMPLIED
    minOccurs     CDATA   #IMPLIED
>
<!ELEMENT primitive ANY>
<!ATTLIST primitive
    id            CDATA   #IMPLIED
    ref           CDATA   #IMPLIED
    maxOccurs     CDATA   #IMPLIED
    minOccurs     CDATA   #IMPLIED
    parent        CDATA   #IMPLIED
    element       CDATA   #IMPLIED
    attributes    CDATA   #IMPLIED
    regexp        CDATA   #IMPLIED
>
```

Figure 4.3: The DTD for JUMBOMarker.

HTML tables for human-friendly display. The process involves the use of a set of templates (consisting of groups of regular expressions) to identify various sections of the output, and processing instructions which specify how to represent the extracted data in XML. The application of stylesheets allows the user to transform the data to other formats for display.

4.1 Design of JUMBOMarker

JUMBOMarker is an XML-based language supporting structured regular expressions for parsing semi-structured documents. The regular expression to match a single line is enclosed in a `primitive` element. The DTD for JUMBOMarker is shown in figure 4.3. Figures 1.13 and 4.4 show extracts from MOPAC and GAMESS output files; there is a lot of implicit structure in the documents and the components are well labelled (an expert human

```
        BASIS OPTIONS
        -------------
        GBASIS=N31          IGAUSS=       6      POLAR=POPLE
        NDFUNC=       1     NFFUNC=       0     DIFFSP=       F
        NPFUNC=       0      DIFFS=       F

        GRID-BASED DFT OPTIONS
        ---------------
        DFTTYP=B3LYP        QOP   =   0.000     PFTYP =NONE
        NRAD  =      96     NTHE  =      12     NPHI  =      24
        NRAD0 =      24     NTHE0 =       8     NPHI0 =      16

        RUN TITLE
        ---------
  j0006057r00@002_631gstar-zmat_005600nsc159980_

 THE POINT GROUP OF THE MOLECULE IS CN
 THE ORDER OF THE PRINCIPAL AXIS IS     1

 THE MOMENTS OF INERTIA ARE (AMU-ANGSTROM**2)
 IXX=    96.956   IYY=   684.856   IZZ=   707.986

 ATOM      ATOMIC                      COORDINATES (BOHR)
           CHARGE         X                   Y                   Z
 O1          8.0    -4.366447       -1.083753        0.5724950
...
 H17         1.0     8.042858       -1.093115       -1.391276

          INTERNUCLEAR DISTANCES (ANGS.)
          ------------------------------
                   O1             C2             O3             C4
  1  O1             0.000      1.346 *    2.259 *    3.553
...
 17  H17            6.648      5.507      5.669      3.252
                   H17
  1  O1             6.648
...
 17  H17            0.000
  * ... LESS THAN  3.000

 TOTAL NUMBER OF BASIS SET SHELLS             =   52
 NUMBER OF CARTESIAN GAUSSIAN BASIS FUNCTIONS =  151
 NUMBER OF ELECTRONS                          =   70
 CHARGE OF MOLECULE                           =    0
 SPIN MULTIPLICITY                            =    1
 NUMBER OF OCCUPIED ORBITALS (ALPHA)          =   35
 NUMBER OF OCCUPIED ORBITALS (BETA )          =   35
 TOTAL NUMBER OF ATOMS                        =   17
 THE NUCLEAR REPULSION ENERGY IS      457.2594492361

        $CONTRL OPTIONS
        ---------------
 SCFTYP=RHF          RUNTYP=OPTIMIZE    EXETYP=RUN
 MPLEVL=       0    CITYP =NONE        CCTYP =NONE
 MULT  =       1    ICHARG=       0    NZVAR =      45    COORD =CART
 ECP   =NONE        RELWFN=NONE        LOCAL =NONE        NUMGRD=    F
 ISPHER=      -1    NOSYM =       0    MAXIT =      30    UNITS =ANGS
 PLTORB=       F    MOLPLT=       F    AIMPAC=       F    FRIEND=
 NPRINT=       7    IREST =       0    GEOM  =INPUT
 NORMF =       0    NORMP =       0    ITOL  =      20    ICUT  =       9
 INTTYP=POPLE       QMTTOL= 1.0E-06
```

Figure 4.4: A section of a GAMESS output file.

```
      MOLECULAR POINT GROUP : C2
      MOLECULAR POINT GROUP : D3h
```

Figure 4.5: The point group of two different molecules as reported by MOPAC.

practitioner could understand what all the numbers mean). Comparing multiple output files from the same program allows the identification of parts of the output as *boilerplate*[*] which does not vary between documents. Figure 4.5 shows the point group reported by MOPAC for two different molecules; from this it is determined that the string:

```
MOLECULAR POINT GROUP   :
```

is boilerplate. The template to extract the point group is shown in figure 4.6. The extraction is a two stage process. Firstly the document is split into chunks; in this case, the point group (which is captured) and everything else (which is discarded). The more boilerplate present, especially where these provide well defined start and end points for each section, the easier the chunking process is. Once the chunk has been identified, whatever is matched within the capture group is available for JUMBOMarker to process. The data in the first capturing group is then recalled using {`$1`}, the second {`$2`} and so on. When run on the text in figure 4.5, this would produce:

```
<cml>
  <job>
    <scalar dictRef='cml:pointgroup'>C2</scalar>
  </job>
  <job>
    <scalar dictRef='cml:pointgroup'>D3h</scalar>
  </job>
</cml>
```

[*]Boilerplate text consists of standard phrases that may be combined or recalled to create new documents; it is commonly used in contracts or other agreements detailing terms and conditions.

```
<template default="job">
<cml>
  <moduleGroup name="job" id="job" maxOccurs="999999">
  <job>
    <primitiveGroup name="pointgroup">
      <primitive name="pointgroup.p" regexp="     MOLECULAR POINT GROUP   :\s+(.*\S)">
        <scalar dictRef="cml:pointgroup">{$1}</scalar>
      </primitive>
    </primitiveGroup>
  </job>
  </moduleGroup>
</cml>
</template>
```

Figure 4.6: The template to extract and markup the point group from MOPAC output.

```
          CARTESIAN COORDINATES
    NO.       ATOM           X             Y             Z
     1         O        0.00210000   -0.00410000    0.00200000
     2         O       -0.06910000    5.24140000    0.03230000

                                          ⋮

    14         H       -3.16080000    1.41050000    0.94380000
    15         H       -3.18530000    1.42060000   -0.83600000
```

Figure 4.7: Section of MOPAC output: cartesian coordinates.

Unforeseen character strings present in the output, such as error messages, usually prevent regular expressions from matching. The grouping of regular expressions into templates dictates that if one does not match then the entire group fails to match. Further processing is dependent on whether or not the template is mandatory. If the template is mandatory (`minOccurs` $\geq$ `1`) the parser produces an error message, discards all the data for the job and moves on to subsequent jobs. In other words any job containing unexpected data is lost. However, if the template is optional (`minOccurs = 0`) the parser attempts to match the next template against the output. This only results in loss of the data associated with the section in which the mismatch occurred.

The removal of the block structure might be considered as a way to avoid discarding mismatched jobs. In such a system, if an unexpected message were encountered, the parser would jump the line containing the message and then

```
                NET ATOMIC CHARGES AND DIPOLE CONTRIBUTIONS
    ATOM NO.   TYPE          CHARGE      No. of ELECS.   s-Pop       p-Pop
      1          O          -0.278945        6.2789     1.86320     4.41574
      2          O          -0.282193        6.2822     1.86399     4.41820

                                              .
                                              .
                                              .

     14          H           0.055649        0.9444     0.94435
     15          H           0.055699        0.9443     0.94430
```

Figure 4.8: Section of MOPAC output: net atomic charges and dipole contributions.

continue trying to match the current regular expression. However, although MOPAC output contains many stock-phrases, an unambiguous individual line recognition technique is not possible. For instance, consider the two types of data shown in figures 4.7 and 4.8; a regular expression to match a numeric line of the `CARTESIAN COORDINATES` would have the following form:

```
\s+DIGIT+\s+CHEM_ELEMENT\s+FLOAT\s+FLOAT\s+FLOAT
```

where:

```
CHEM_ELEMENT = (H|He|Li|Be|...|Uuo)
DIGIT = [0-9]
FLOAT = (\+|-)?DIGIT+\.DIGIT+
```

This regular expression would also match numeric lines in the `NET ATOMIC CHARGES AND DIPOLE CONTRIBUTIONS` block, if the element in question has only *s* electrons. The only way of removing the ambiguity between the two lines is by context. Context is primarily provided by textual headings and position in the document. The only way to determine context in such documents is by chunking, hence the block structure of JUMBOMarker is necessary.

4.2 JUMBOMarker: Single-Pass, Single-Parse

The original design of JUMBOMarker consisted of a single-pass, single-parse parser; this means that the input document is only read once (single-pass) and that the data is extracted in a single process (single-parse). This design

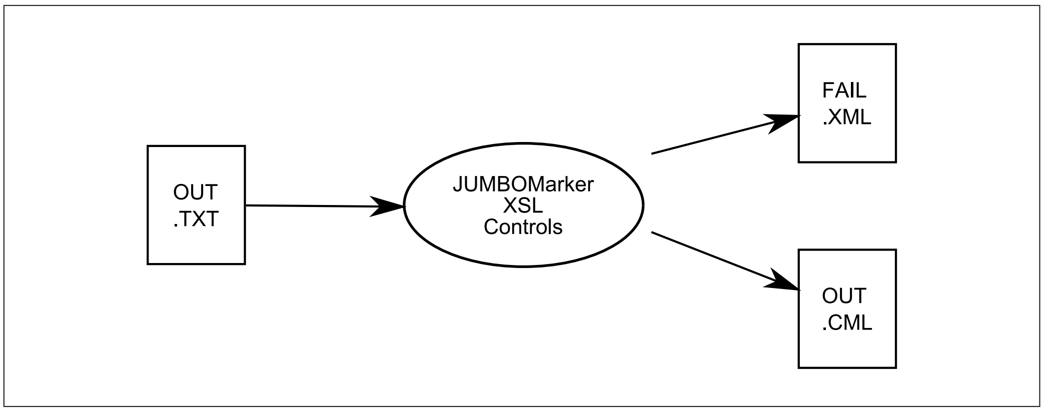


Figure 4.9: For single-pass, single-parse parsing JUMBOMarker is an overarching program that determines what output should be created, all the temporary files being held in memory.

(shown in figure 4.9) resulted in an overly large, complex and extremely fragile system. Much of the complexity was a result of the need to control which routine should be run as a result of the presence or absence of particular matched groups in the output document (held in memory) before the final files are created. Because the templates to parse each computational code differ, specific methods are required to control the output for each. Therefore, to adapt JUMBOMarker to parse the output from a different computational chemistry program required editing the source code and effectively creating a new program.

The single-pass, single-parse approach yielded ca. 95% correct parsing. All the incorrectly-parsed documents resulted from the presence of unexpected strings in the output. To improve and simplify the parsing process, a two-pass, two-parse parser was introduced. This had the added benefit of separating the various components of the program. The separation on the components resulted in a more robust program and greatly simplified the process of extending JUMBOMarker to parse other program output.

4.3 JUMBOMarker: Two-Pass, Two-Parse

In the two-pass, two-parse approach, JUMBOMarker is only used as a parser. In effect this implementation uses two single-pass, single-parse parsers which are controlled by an external program (figure 4.10). The first of these parsers is used to determine whether or not the job has failed. If the job has failed, the second parser is not used. However, if the job has run successfully, the control program runs the second parser to extract the data. Although this description refers to two separate parsers, they are in actuality the same parser (JUMBOMarker) using different templates.

The first parser runs JUMBOMarker over the output log file with the failure template. The failure template serves two purposes, to determine that the output log file is complete and to check that there are no error or fail messages in the log file. Each of the `primitiveGroups` have the `minOccurs` attribute set to `0` and `maxOccurs` set to `1`. Thus an XML file is always produced — even if it only contains an empty root element. A stylesheet is then run over the XML files produced to determine if the job is complete and contained no error messages. If this is the case, the output log file is suitable for the further parsing. If it is not, a further file (*fail.xml*) containing the failure reason is created.

There are two sources of failure; incomplete output files and programmatic errors. If the failure reason is that the output file is incomplete then the job can simply be resubmitted for calculation. Jobs which cause programmatic errors are analysed and the appropriate refinements are made to the protocol (see chapter 7). The list below shows the sections present in a GAMESS failure template:

- gamess-begun
- too-many-steps
- wrong-charge-and-multiplicity

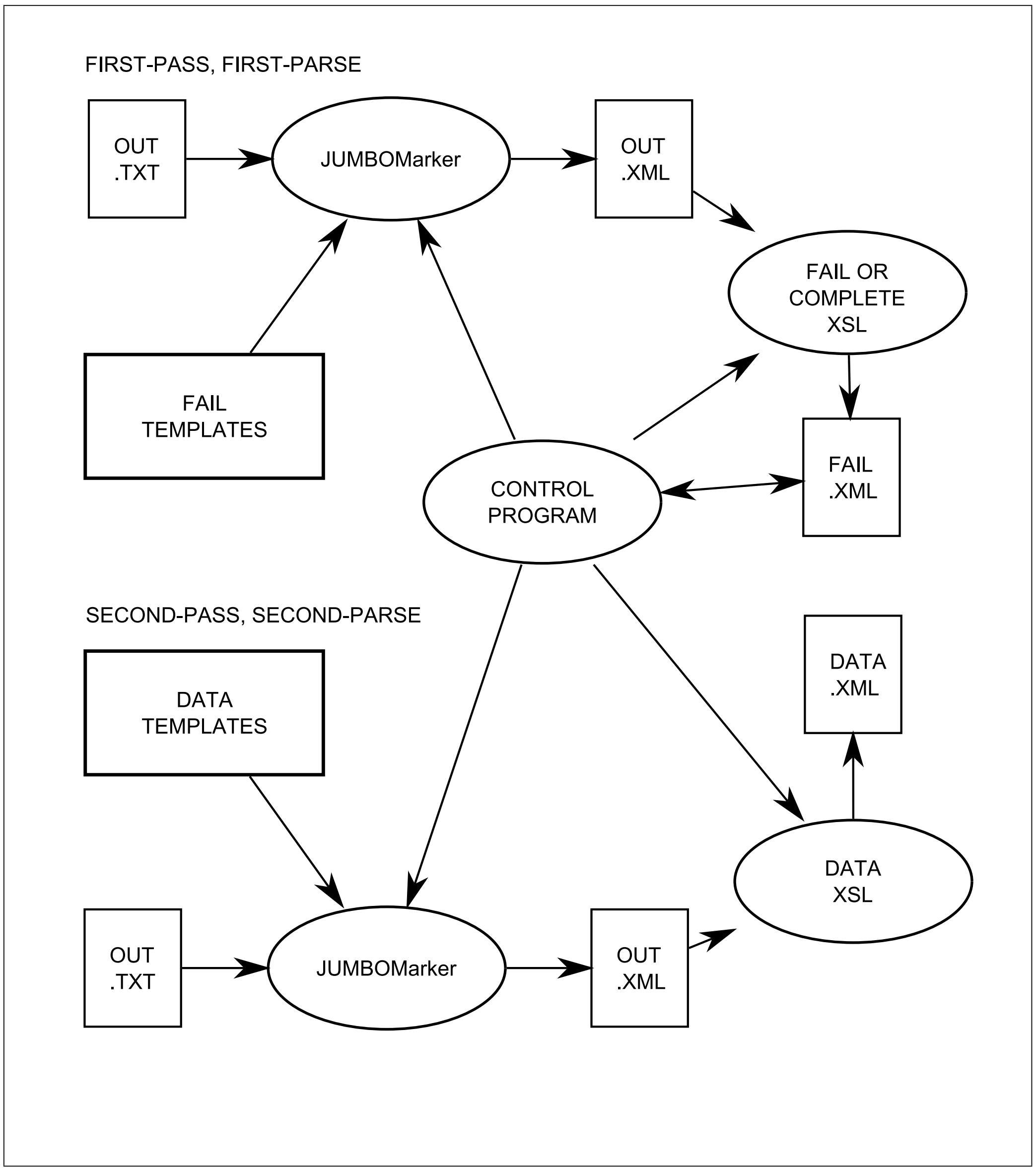


Figure 4.10: JUMBOMarker is only used to parse the document, each step is controlled by external programs or scripts. The second parse is dependent on the results of the first.

- scf-not-converged
- too-little-time-to-do-another-point
- atoms-too-close
- general-error
- general-failure
- gamess-terminated
- error-termination

The presence of both the `gamess-begun` and `gamess-terminated` sections indicate that the entire document is likely to be complete. If any of the other sections are also present this indicates that the job has failed for an identifiable programmatic reason. If there is no failure, the control program then applies the second parser to extract the data.

The introduction of the two-pass, two-parse JUMBOMarker greatly simplified the detection of errors in the log files and hence the parsing process. Whilst this system was sufficient for the extraction of the required data from MOPAC output, it was incapable of fully parsing a GAMESS output file because of the repeating groups present in these files. To address this issue a multi-pass, multi-parse approach was adopted.

4.4 JUMBOMarker: Multi-Pass, Multi-Parse

The limitations of the two-pass, two-parse approach are those of ordering and choice. JUMBOMarker can parse the following document structure:

```
A* B? C+
```

where `A`, `B` and `C` are all sections of the document. However, it does not support:

```
( A | B ) C ( D & E ) F
```

where D, E and F are all sections of the document and the ampersand means *at least one of each but in any order.* It is in general unnecessary for the ampersand connector to be supported to parse output files and the ‘or’ connector (whilst useful) is not vital provided that the type of output is known. For example, different (though largely very similar) templates can be created to parse the output from different run types.

By using optional repeating groups JUMBOMarker can parse documents with the structures:

```
A B A B A B A
A A A B B B A
B B A A A B B B A
```

where A and B represent primitives within a primitiveGroup (with maxOccurs > 1). However, there cannot be any unparsed data between A and B. Thus documents of the form:

```
A C B A C B
```

where C represents a section of the document for which no template exists cannot be parsed. All quantifiers in JUMBOMarker are greedy, that is, they match as much as they possibly can; thus there is no way of ‘skipping’ over a section because a general template to match an unspecified number of lines (`<primitive regexp=‘.*’ minOccurs=‘0’ maxOccurs=‘unbounded’ />`) would continue to match to the end of the document and B would never match.

The introduction of multi-parse parsing eliminates this problem. Figure 4.11 shows the multi-parse process. The first parse through the document

```
A C B A C B
```

would markup the A sections producing

```
Ã Ã
```

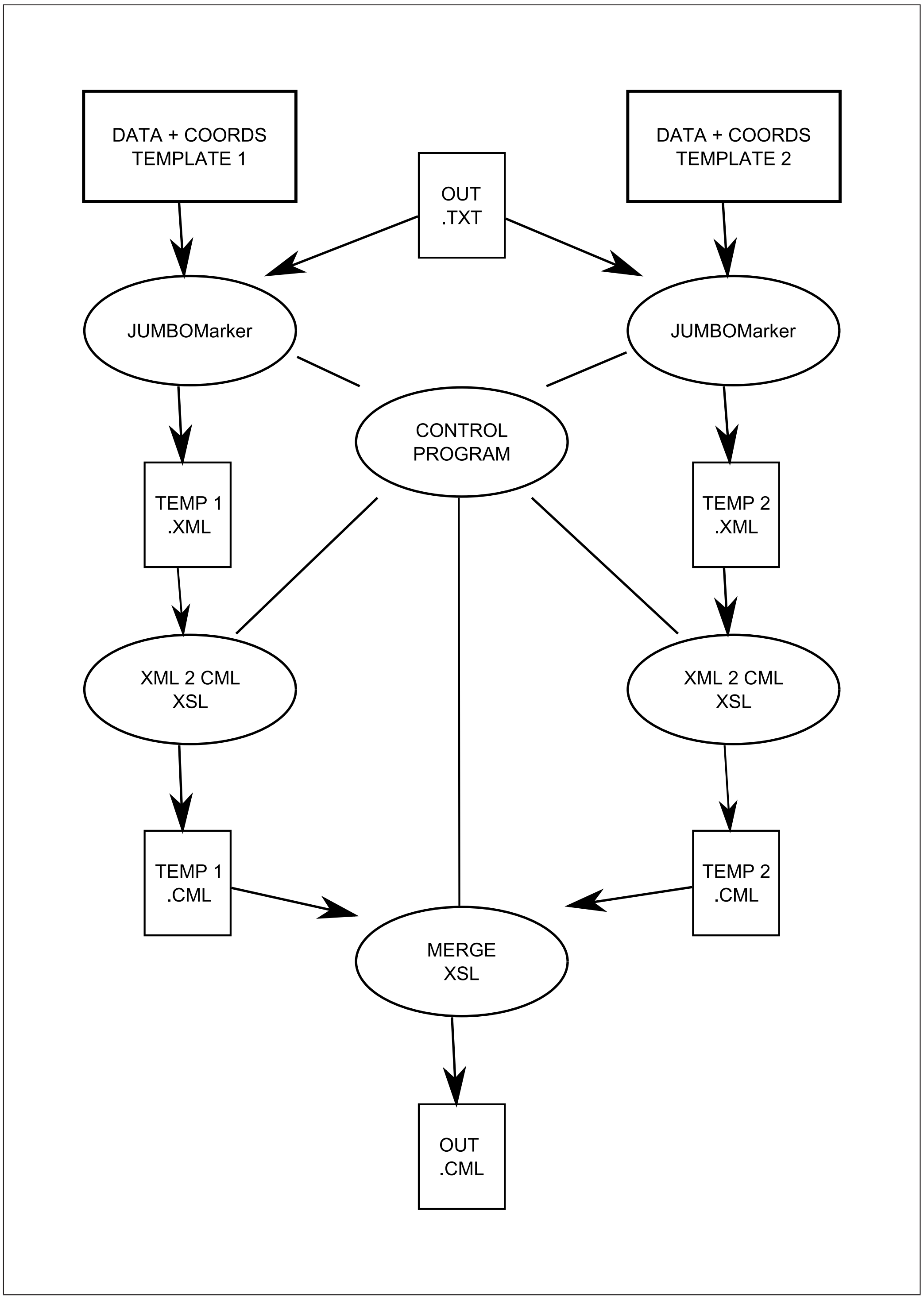


Figure 4.11: The out.txt will only be parsed if it has already passed the failure test. In multi-pass multi-parse parsing JUMBOMarker traverses the input file multiple times, each time producing a different file — this is necessary for overlapping repeating groups.

(where `Ã` represents the XML markup of section `A`). The second parse would mark up the `B` sections producing

`B̃ B̃`

These sections would then be merged using stylesheets to produce

`Ã B̃ Ã B̃`

This version of JUMBOMarker implements the same initial single-pass, single-parse parser as that in the two-pass, two-parse version. However, if the job has not failed the extraction of the data is no longer performed in a single-process but by multiple parsers, the results of which are then combined to give all the extracted data.

The multi-pass, multi-parse approach took an average of five seconds to check that a file had not failed, extract the data and coordinates and extract the times reported into CML — this figure includes the initial loading time for the program and all the reading and writing to disk that is necessary. The parsing was performed on a Sony VAIO laptop computer; Intel Pentium 4 2.8 GHz cpu with 512 Mb of RAM.

4.5 Conclusions

The current multi-pass, multi-parse version of JUMBOMarker achieves greater than 99.9% correct parsing of all files with no human curation. The files which are not correctly parsed are simply not parsable without a complete rewrite of the underlying code and might still require human input. The main problem remaining is that of encoding particular characters in the XML representation. This is a general issue for any XML based system [143]. Fortunately, this problem occurs only rarely — only one instance was found during the course of this work. The extremely low error rate of JUMBOMarker allows it to be incorporated into automated processes thereby reducing the amount of work to be done by humans. The use of JUMBOMarker as part of a workflow to parse program output for analysis forms an integral part of the following chapters.

Although very little human intervention is required once JUMBOMarker has been set up, the creation of the templates still requires a large amount of user input and it is desirable to reduce this. Methods to automatically generate templates, or parts of templates which could be fleshed out by the user, were examined (using HMMs and extended natural language parsing techniques). However, whilst such techniques could provide a rough outline of the required template a large amount of *debugging* was required before they were usable. In fact, because the user was not involved in the initial creation of the templates, it was often found that more time was required to work out what was required than if these methods were not employed. In general, it proved simpler and faster to create the templates by hand. Collaborations with Martin Braendle [144], René Kanters [145] and Miguel Howard [146] have led to the creation of templates for *GAUSSIAN03* [147] output and the implementation of the JUMBOMarker approach to parse user-defined extra data for programs such as Jmol.

Chapter 5

High-Throughput Computing

High-Throughput (HT) computing involves allowing users to run large numbers of independent jobs simultaneously over long periods of time. Conversely, High-Performance computing is aimed at provided large amounts of computing power for relatively short periods of time. HT computing is particularly applicable to the process of optimising molecular geometries — the same algorithm is applied repeatedly to different molecules independently.

A typical computer is used very inefficiently; it is often idle for a very large percentage (>80%) of its available cycles. Various groups have attempted to to make use of these idle cycles — such as the Berkeley students behind the SETI@home [148] project and CANCER@HOME [149]. However, these schemes have all involved users downloading a purpose-built program which runs whilst the computer would otherwise be idle. These systems only allow a specific program to be run. Therefore a group at the University of Wisconsin-Madison developed Condor; once installed on a machine this allows many (any) programs to be run whilst the machine would otherwise be idle [150].

5.1 Condor

The idea behind Condor is simple; it matches any computational jobs that users have with spare power in other owners' computers. Computer owners do not have to modify their programs to use Condor, they just have to agree to become part of a Condor network. This network is a group of computers connected in such a way that messages and data can pass between them. Condor remains in the background of the network, on a computer called the central manager — which can be any computer in the network — where it searches for inactive computers. When such a machine is found, Condor claims it and adds it to the list of available nodes. By default, a physical user has higher priority than Condor so when the owner resumes using the computer, Condor stops any currently running programs and removes the computer from the available nodes list.

In practice Condor is more complicated. Every computer in the pool must continually run parts of the Condor software that track activity on both the central manager and the local system. To do this, the computer must be capable of *multi-tasking*, or running more than one piece of software simultaneously. The overall effect is that by using Condor previously unusable computing resources and time become available for free.

Condor implements an internal scheduling system so that large jobs do not drain the pool of cycles, this is based on the priority Up-Down algorithm developed by Livny *et al.* [151]

> This system makes it possible for scientists who are running time-consuming research to coexist with people running shorter jobs. The priority that the system assigns to a scientist's project decreases as the number of cycles the scientist uses increases. [152]

The Unilever Centre for Molecular Science Informatics (UCC) allowed Zhang of the Murray-Rust group to establish a Condor pool on 24 teaching machines in 2002. This provided approximately three months of uninterrupted run time over the summer vacation (6 cpu years), effectively using the Condor system as a scheduler. During term (when interruptions may occur as physical users claim the machines) this reverted back to an idle-cycle scavenging system as per the original design.

5.2 MOPAC and NCI HT Computing

MOPAC is a general-purpose semi-empirical molecular orbital package for the study of solid state and molecular structures and reactions. Semi-empirical Hamiltonians are used in the electronic part of the calculation to obtain molecular orbitals, the heat of formation and its derivative with respect to molecular geometry. Using these results MOPAC calculates the vibrational spectra, thermodynamic quantities, isotopic substitution effects and force constants for molecules, radicals, ions, and polymers.

The National Cancer Institute (NCI) make available the connection tables of 250,251 structures for download and re-use [153]. As described by Zhang in 2004 [154], these structures were downloaded in SDF format and converted to CML using OpenBabel [155], the resultant CML was combined with a set of job controls to create a MOPAC input file. The input file specified the type of calculation to run; a geometry optimisation using PM3 and the charge on the molecule. Where there were multiple molecules in the MOL file the largest fragment was selected*. The Condor submission node does not support 250,251 individual jobs and therefore the input files were collected into 500 groups of 500 and a single group of 251 inputs. These were then submitted to the Condor infrastructure for calculation†. Figure 5.1 shows the overview of this process.

The structure and properties determined by MOPAC were extracted and parsed to CML using JUMBOMarker, combined with the input structure previously generated (thus already in CML) and stored in an XML database (Xindice [156]). This process was non-trivial. Whilst MOPAC is in general a very stable program, the large number of jobs revealed some problems that were previously not encountered. An example of such a problem was a memory overflow issue that affected every 64,000th job, causing it to crash.

*The largest fragment was taken to be that which contained the most atoms.

†The calculation and analysis of the NCI molecules using MOPAC was a collaborative project between Zhang and this author. The submission of the jobs was entirely the work of Zhang, the subsequent analysis the work of this author.

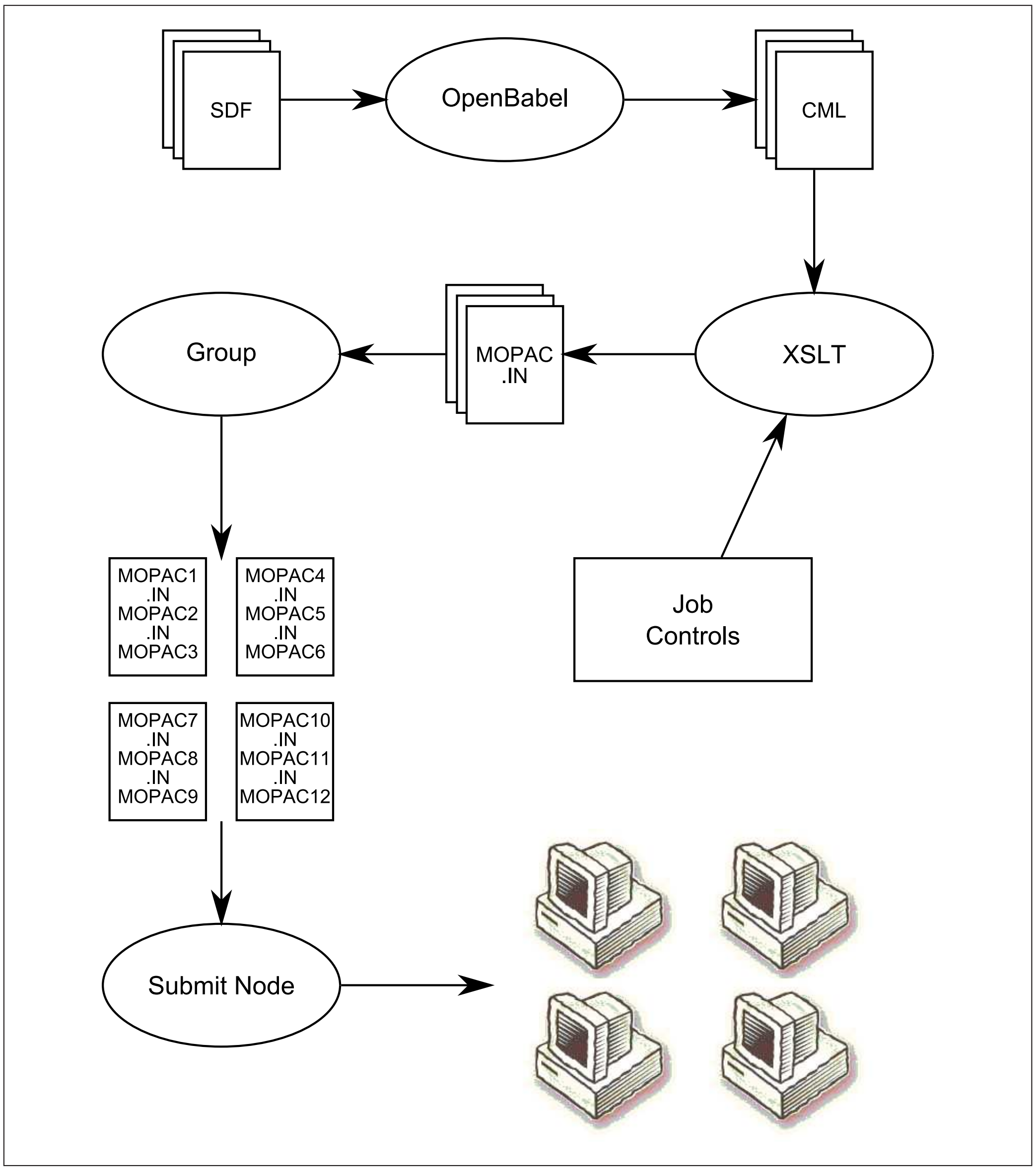

SDF
OpenBabel
CML
XSLT
MOPAC .IN
Group
Job Controls
MOPAC1 .IN MOPAC2 .IN MOPAC3
MOPAC4 .IN MOPAC5 .IN MOPAC6
MOPAC7 .IN MOPAC8 .IN MOPAC9
MOPAC10 .IN MOPAC11 .IN MOPAC12
Submit Node

Figure 5.1: Overview of the MOPAC calculation of the NCI molecules.

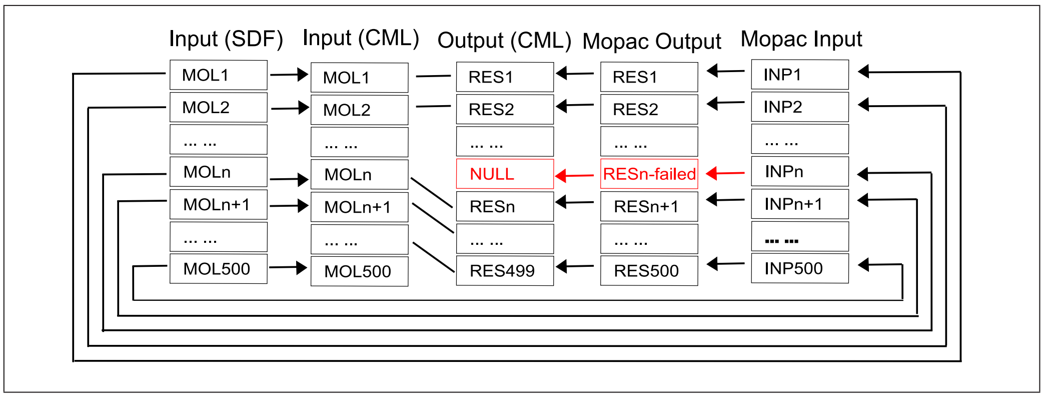


Figure 5.2: Mismatches in the MOPAC input and output

The JUMBOMarker used for this process was the single-pass, single-parse design as defined in section 4.2. The templates developed for this process were designed to be fault tolerant — that is, if a job failed JUMBOMarker should skip over this job and continue parsing. Hence, the simple matching system for input and output files often resulted in mismatches as shown in figure 5.2.

The detection of this fault was relatively trivial because the matching program reported the mismatch, for example;

> no output structure was found for input molecule 500.

The fault was not easy to rectify because the files did not contain unique identifiers. Hence determining which was the appropriate output file to match with a particular input file relied on matching the structures contained in each — InChIs are ideal for this purpose.

5.2.1 InChI

InChI is the IUPAC International Chemical Identifier and is a string of characters that is designed to canonically represent a chemical substance [17, 157]. It is derived from a structural representation of the substance in a way designed to be independent of the structural representation. Therefore a single

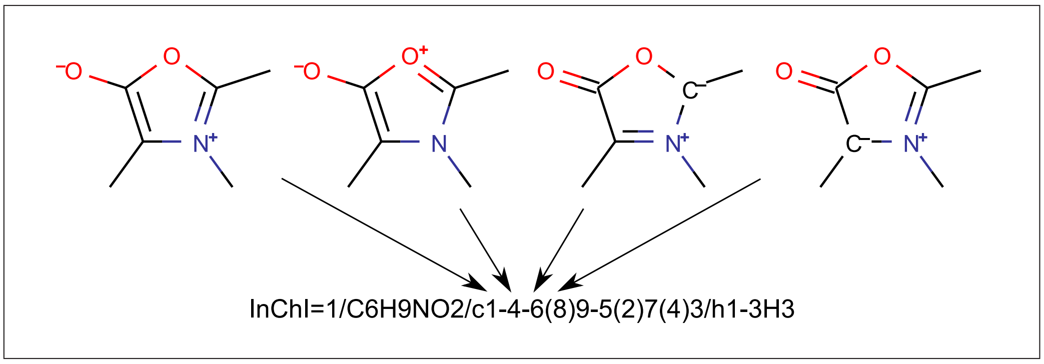


Figure 5.3: Various structures that all are represented by the same basic InChI.

compound will always produce the same basic identifier. Figure 5.3 shows various structural representations of the same molecule which are normalised to a single basic InChI.

The InChI technical manual states:

> It was agreed at IUPAC meetings prior to the start of this project that the first version of the InChI should cover well-defined, covalently-bonded organic molecules. It was also agreed to include substances with mobile hydrogen atoms (tautomers, for instance). In the course of this project, it was found that straightforward extension organometallic compounds could be represented. Methods were found to also include variable protonation. Also, the present version only considers traditional organic stereochemistry (double bond — sp^2 and tetrahedral — sp^3) and the most common forms of H-migration (tautomerism). However, the layered structure of the InChI allows future refinements with little or no change to the layers currently used. [17]

It is possible to generate InChIs for molecules that the program was not designed for and those that are not chemically valid, for example $C(CH_3)_6$ generates

InChI=1/C7H18/c1-7(2,3,4,5)6/h1-6H3

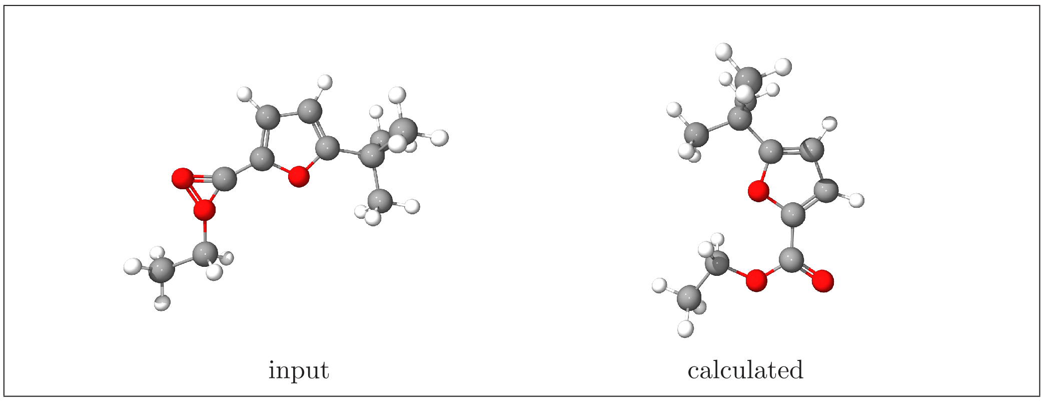


Figure 5.4: Garbage-In, Good-Out: a molecule from the NCI that was *mended* by MOPAC.

but with the warning

```
message:  type=warning value=Accepted unusual valence(s):  C(6)
```

The use of InChIs to index and identify documents (especially web-based documents) was described by Coles *et al.* in 2005 [158]. It was hoped that creating InChIs for the structures contained in both the input and output documents should allow the correct molecules to be matched up, unfortunately this did not always prove to be the case. In several cases the connection table (from which the InChI is derived) changed as a result of the MOPAC calculation. The term *proteus* was coined to describe this type of molecule[‡]. It is interesting to note that occurrence of proteus molecules in the dataset would likely have remained undiscovered if the mismatching problem had not occurred.

5.2.2 Proteus Molecules

Analysis of the proteus molecules showed that in some cases the MOPAC calculation had *mended* a *bad* input structure (*Garbage-In*, *Good-Out*) whilst in other cases the MOPAC calculation *broke* a *good* input structure (*Good-In*,

[‡]Proteus was a shape-changing ocean deity in Greek mythology.

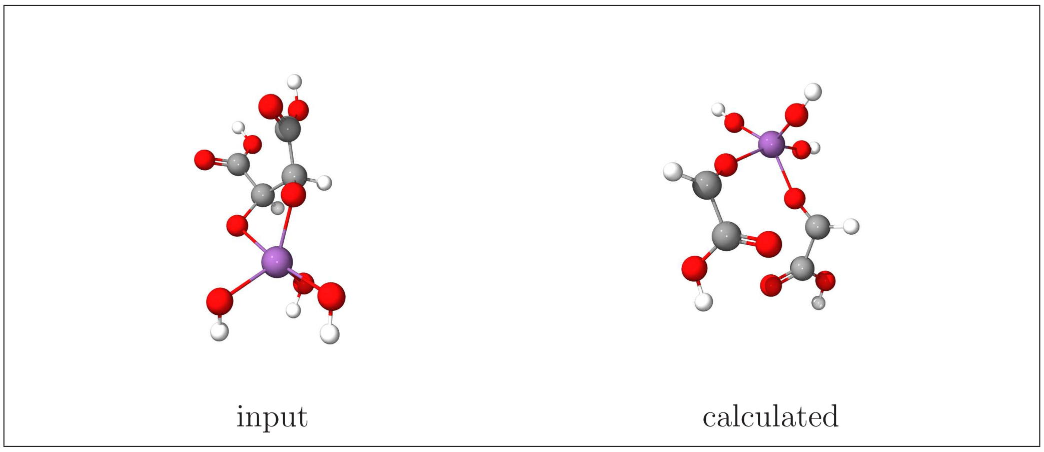


Figure 5.5: Good-In, Garbage-Out: a molecule from the NCI that was *broken* by MOPAC — the five coordinate atom is antimony.

Garbage-Out). Examples of these are shown in figures 5.4 and 5.5 respectively. In general the Good-In, Garbage-Out molecules were as a result of attempting to calculate properties for molecules that MOPAC is not optimised to deal with (under the protocol that was in use), *i.e.* those containing heavy metals. The identification of *good* and *bad* structures might be possible using sufficiently chemically-aware programs but currently is performed by human experts.

As previously discussed, it is important to verify that data is of sufficient quality before it is publicly disseminated. Determining which of the structures calculated were Good-Out (regardless of whether or not input structure was good) is difficult because of the inconsistent quality and volume of the NCI dataset. To provide a measure of the quality of the MOPAC structures, comparison with structures obtained from a different source is necessary (figure 5.6). Ideally the comparisons between structures would be between the molecules in the MOPAC calculated set and the same molecules determined experimentally.

Experimentally-determined 3D structures are available from X-ray crystallography reported in Crystallographic Information Files (CIFs). The format

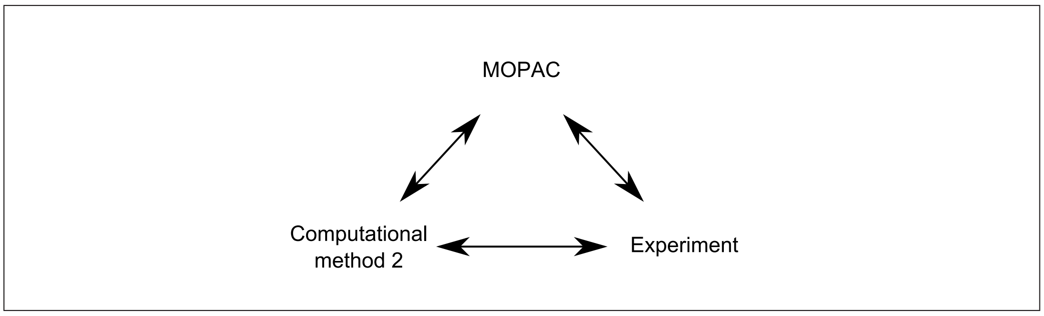


Figure 5.6: To cross-check the MOPAC results the structures can be compared to experimentally-determined structures or those determined by a different computational program

of a CIF is discussed in chapter 6 and the processing necessary to extract the CTs, in section 7.3. For the moment it is sufficient to state that the extraction of the CTs is non-trivial and in general, not possible without downloading the entire CIF. The lack of the CT, or an identifer directly derived from it (such as InChI), in an immediately machine-understandable form, means that it is impossible for a machine to automatically search the literature to find appropriate structures for comparison. It was decided that it would be more tractable to compare the geometries determined by MOPAC with those calculated by a different program at a higher level of theory.

5.3 MOPAC and GAMESS

GAMESS (General Atomic and Molecular Electronic Structure System) allows the optimisation of molecular geometries using the energy gradient (calculated analytically for SCF or DFT wavefunctions) [159]. The original code split in 1981 into two variants, GAMESS (US) and GAMESS (UK) which now differ substantially. This work has been performed using GAMESS (US) which is available at no cost to both academic and industrial users. This work investigates the behaviour of GAMESS in a HT environment and uses the results to validate the results of the MOPAC optimisations. To do this, a *general purpose* protocol was required — the protocol should be able to cope with a wide range of different molecules and still converge the SCF and optimise the geometry.

GAMESS input is directed by what is called '`$`' or '`$ groups`'. The input components directing what type of calculation should be performed are included in the `$CONTRL` group. The `$DATA` section provides the specific molecular set up, including both the symmetry type and the 3D coordinate information necessary to construct the molecule. This can be provided as a set of Cartesian components or, as a `Z-matrix` which describes the molecules using internal coordinates. Figure 5.7 shows a typical input file.

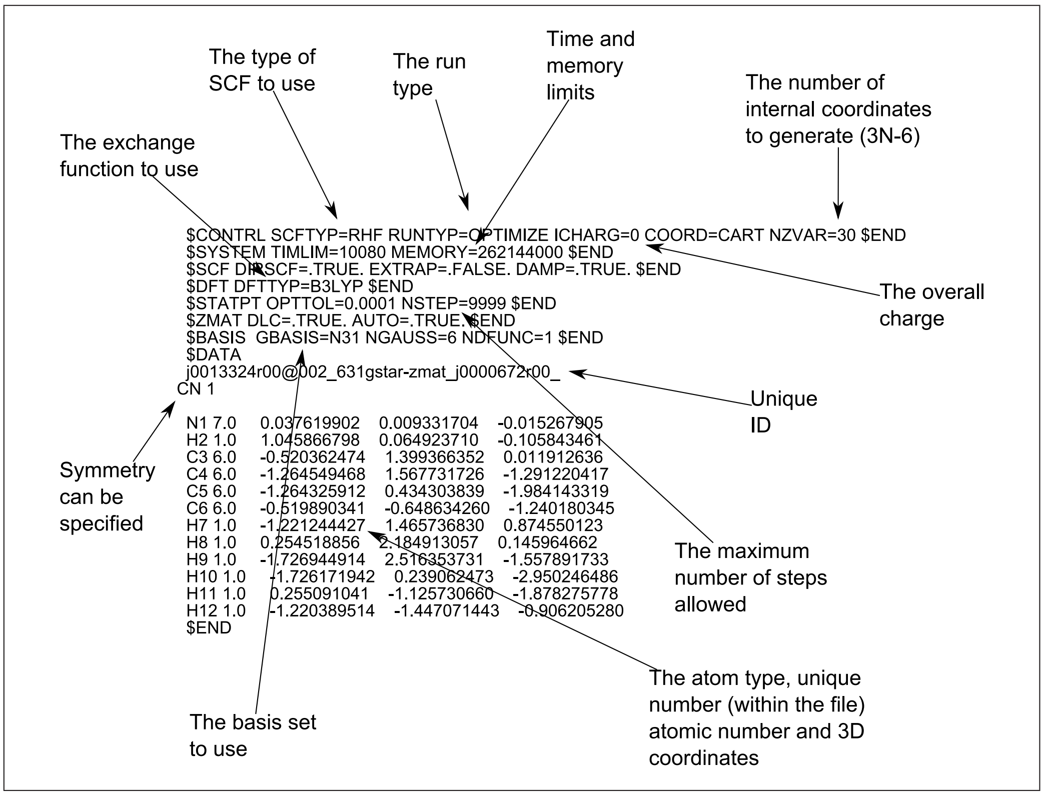
Whilst the protocol was designed to reduce the number of failures, some failed jobs were anticipated. The failures were expected to fall into four categories:

System crashes The calculation can either be regarded as lost or resubmitted for calculation.

Calculations overrun The protocol can be adapted to give more time for each job (the time is effectively free) but this might be a sign of a pathological molecule which will always cause problems.

The *wrong* answer is produced There are four possibilities;

- The protocol is not capable of dealing with molecules of this type (for example open shell systems). Either filters should be implemented to remove molecules of this type or the protocol should be changed to accommodate them.
- This is a general problem (everyone gets the same wrong answer), as such this should not be considered a problem because the protocol should be developed to get consistent answers rather than *right* ones.
- The result produced by GAMESS is different from that from other programs — such results are a GAMESS community concern and would be reported to the developers.



```
$CONTRL SCFTYP=RHF RUNTYP=OPTIMIZE ICHARG=0 COORD=CART NZVAR=30 $END
$SYSTEM TIMLIM=10080 MEMORY=262144000 $END
$SCF DIRSCF=.TRUE. EXTRAP=.FALSE. DAMP=.TRUE. $END
$DFT DFTTYP=B3LYP $END
$STATPT OPTTOL=0.0001 NSTEP=9999 $END
$ZMAT DLC=.TRUE. AUTO=.TRUE. $END
$BASIS  GBASIS=N31 NGAUSS=6 NDFUNC=1 $END
$DATA
j0013324r00@002_631gstar-zmat_j0000672r00_
CN 1

N1 7.0     0.037619902    0.009331704    -0.015267905
H2 1.0     1.045866798    0.064923710    -0.105843461
C3 6.0    -0.520362474     1.399366352    0.011912636
C4 6.0    -1.264549468     1.567731726    -1.291220417
C5 6.0    -1.264325912     0.434303839    -1.984143319
C6 6.0    -0.519890341    -0.648634260    -1.240180345
H7 1.0    -1.221244427     1.465736830    0.874550123
H8 1.0     0.254518856     2.184913057    0.145964662
H9 1.0    -1.726944914     2.516353731    -1.557891733
H10 1.0     -1.726171942    0.239062473    -2.950246486
H11 1.0     0.255091041    -1.125730660    -1.878275778
H12 1.0     -1.220389514    -1.447071443    -0.906205280
$END
```

The type of SCF to use
The run type
Time and memory limits
The number of internal coordinates to generate (3N-6)
The exchange function to use
The overall charge
Unique ID
Symmetry can be specified
The maximum number of steps allowed
The atom type, unique number (within the file) atomic number and 3D coordinates
The basis set to use

Figure 5.7: An example GAMESS input file. The molecule specified in cartesian coordinates (`COORD=CART`) should be geometry-optimised (`RUNTYP=OPTIMIZE`) using automatically-generated internal coordinates (`$ZMAT` group). The maximum time (`TIMLIM`) allowed for the calculation is 10080 minutes (one week) and a maximum of 262144000 8-byte words of memory are to be used (`MEMORY`). The energies are calculated using the Restricted Hartree Fock wavefunction (`SCFTYP=RHF`) theory with the B3LYP exchange function (`DFTTYP=B3LYP`) optimisation. Pople's N-31G split valence basis set is to be used (`GBASIS=N31`) available for H-Zn when NGAUSS=6 and one polarisation function to be included on atoms of atomic number 3 and over (`NDFUNC=1`). The basis set and exchange function were chosen because they represent the best trade-off between calculation time and accuracy in common use [160].

- The result produced by this system differs from that calculated by another GAMESS system using the same level of theory. This is probably the most important wrong answer to identify, but does not necessarily mean that the answer is wrong; for example different starting geometries may lead to different local minima. Repeating calculations of the same molecule on different systems (in this case Linux- and Windows-based architectures) allows the determination of *acceptable* variation.

Unforeseen failures Although every attempt is made to predict how calculations might fail, previous experience suggests that unforeseen failures do occur. These are often useful because they highlight new ways of looking at the data and adapting the protocol.

Chapter 7 details how a protocol is created and refined which involves choosing an appropriate level of theory and how to determine which structures are suitable for calculation at that level. However, the method presented relies on an assumed knowledge of the behaviour of the calculation program. The initial creation of a suitable protocol for HT computing with an unfamiliar program is extremely difficult and the first attempt should be expected to fail. The development cycle relies on the detection of problems to determine problems with the protocol which can then be refined or filters put in place. Figure 5.8 shows a typical protocol development cycle.

The intention is for humans to validate protocols rather than individual data items (when large datasets are being used, dealing with individual data items is often impossible). Thus when developing a protocol, a molecule that is correctly computed is of no immediate interest because it does not highlight any problems present. Once the protocol has been established all the molecules that were correctly computed using that protocol can be analysed to produce derived data and in some cases further refinements for that protocol.

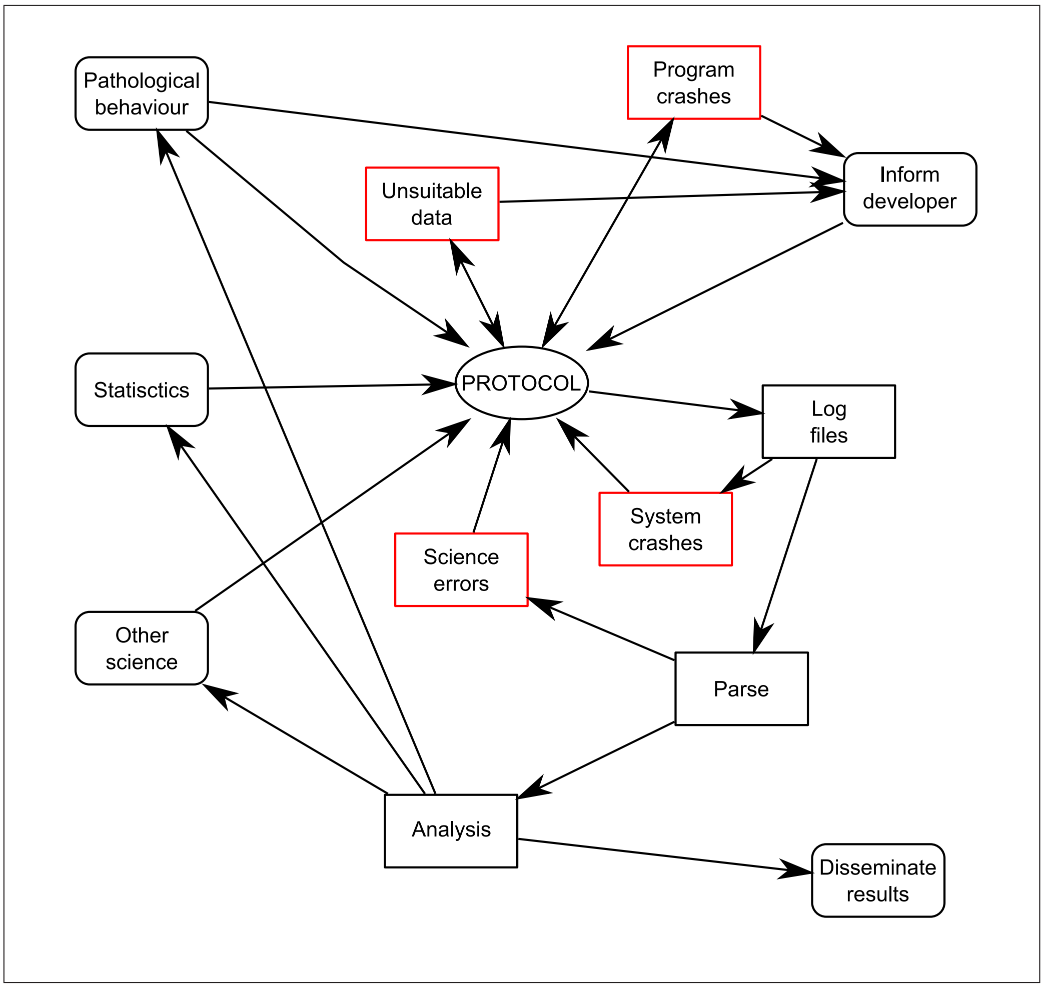


Figure 5.8: A typical development cycle for a protocol

Typical examples of the types of error encountered during the protocol development and the resultant action taken are described below:

System crash A thunderstorm caused a power cut that prevented the air conditioning working, thus the computers overheated and stopped; all affected jobs were restarted.

Science error A large number of the results failed because the incorrect charge was specified; a bug was found in the code that created the input file which resulted in all molecules being submitted with an overall charge of zero. The code was corrected and all the jobs were resubmitted.

Unsuitable data Open shell systems were being submitted for calculation to a protocol for closed shell systems only; a filter was created to prevent open shell systems being submitted.

Program crash Partial log files were produced for some results, the cause of this remains unknown; the affected jobs were resubmitted.

Further refinements to the protocol resulted from analysis of the parsed data.

To increase the number of molecules that can be calculated, it is desirable to reduce the calculation time for each one. The most obvious way to do this for molecules that contain symmetry is to only calculate the symmetry unique atoms. Unfortunately this can lead to problems:

> When the point group contains a 3-fold or higher rotation axis, the degenerate moments of inertia often cause problems choosing correct symmetry unique axes, in which case you must use `COORD=UNIQUE` rather than Z-matrices. Warning: The reorientation into principal axes is done only for atomic coordinates, and is not applied to the axis dependent data in the following groups: `$VEC`, `$HESS`, `$GRAD`, `$DIPDR`, `$VIB`, nor Cartesian coords of effective fragments in `$EFRAG`. `COORD=UNIQUE` avoids reorientation, and thus is the safest way to read these. [161]

	no Z-matrix	Z-matrix
Total time (s)	12090251	8124016
Mean time per molecule (s)	17650	11860
Mean time per basis function (s)	154	104
Mean time per non-H atom (s)	2752	1849

Table 5.1: Comparing the effects of not implementing and implementing internal coordinates (Z-matrix) on the calculation time. The statistics are derived from the successfully completed geometry optimisations of 685 molecules containing between three and eight non-hydrogen atoms.

It was felt that this would not be a worthwhile time-saving approach, largely because it would require many man-hours to create the new protocol and because the resultant protocol would be less general.

Other methods are available to reduce the calculation time (lowering the level of theory for instance) but in general:

> ... geometry optimizations depend on the coordinates being used (not the starting values, but rather the type). In general the most satisfactory behavior for the least human effort comes from putting cartesian coordinates in `$DATA`, selecting `NZVAR=3N-6`, and then using `$ZMAT DLC=.TRUE. AUTO=.TRUE. $END`. [162]

The creation of the internal coordinates is automated within GAMESS [163] but may fail:

> ... in cases where the automatic coordinate generation fails, you will have to play with the `NONVDW`, `IXZMAT`, and `IRZMAT` keywords in the same group. In addition, you always have to set the keyword `NZVAR` in the `$CONTRL` group to $3N - 6$ (or $3N - 5$ in case of linear molecules) when you use the DLC option, with N being the number of atoms. [164]

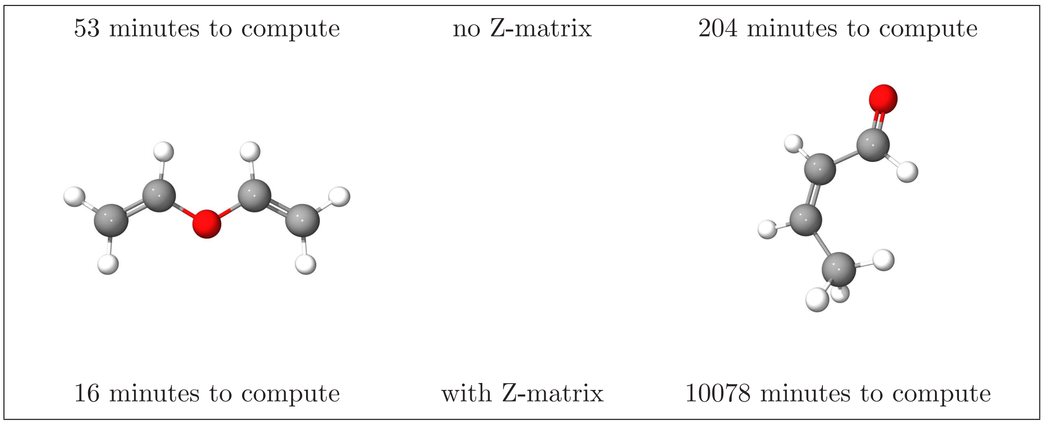


Figure 5.9: Although in general using a Z-matrix reduces the calculation time, this was not always found to be the case. Both molecules were undergoing geometry optimisation using 6-31G*/B3LYP.

The programs used to create the input file contained no methods for ascertaining the linearity of a molecule therefore the `NZVAR` was always set to $3N - 6$. In fact, no linear molecules that were suitable for calculation have been found in the dataset to date.

In general the calculation time was found to decrease when a Z-matrix was used (see table 5.1), this was not the case for all the molecules. Figure 5.9 shows the effects of using internal coordinates on two structural isomers. Whilst such cases may be interesting on an individual basis, from the point of view of the protocol developer they are exceptions to the general case and have little overall effect. To quote one of the developers of GAMESS:

> ...it is possible for some molecules, some times, to work better in other coordinates. This is numerical work, and exceptions always turn up. [162]

The possibility of a small number of automatic coordinate generation failures was deemed an acceptable trade-off for the general decrease in calculation time, thus a Z-matrix was implemented in subsequent protocols.

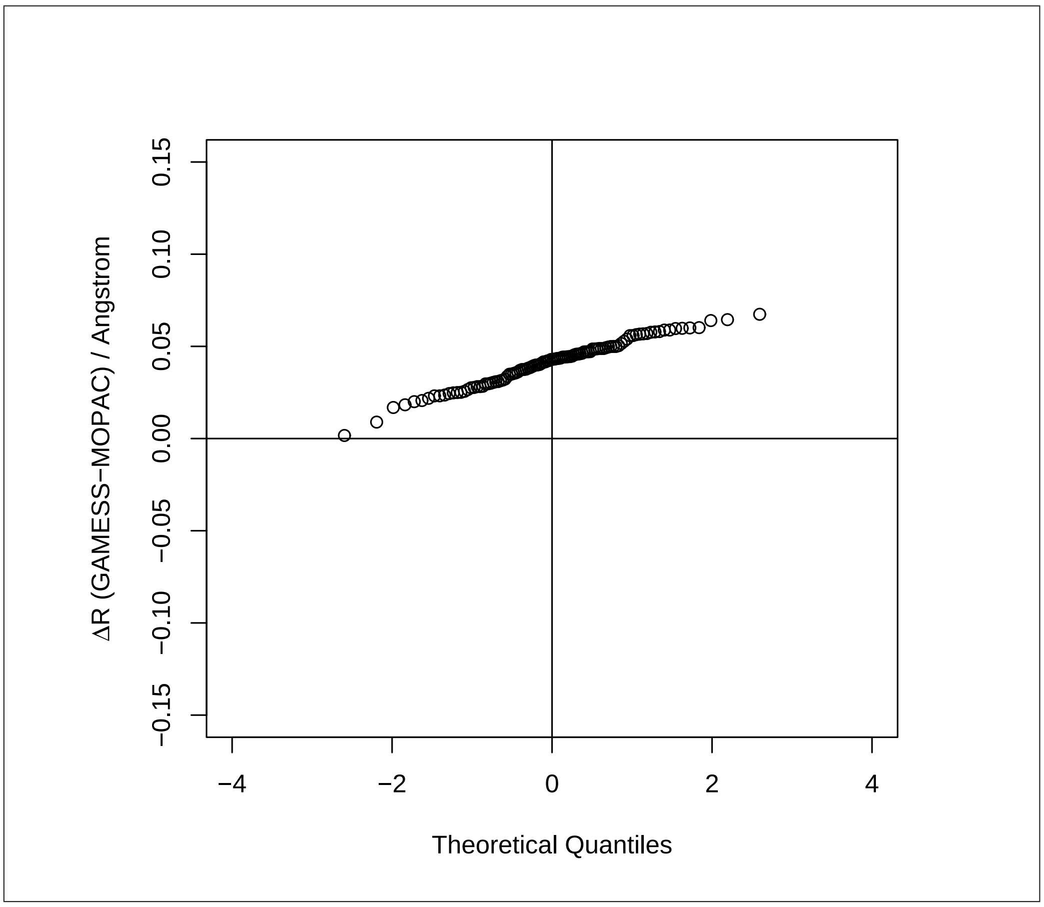


Figure 5.10: QQ plot for the 106 C–Cl bonds in the dataset — the data are normally distributed over the entire length with $\overline{\Delta R}$=0.041Å and s=0.013Å.

5.4 Results

The difference in bond length (ΔR(GAMESS$-$MOPAC)) is expected to be normally distributed; this can be assessed graphically by using Quantile-Quantile plots (QQ plots). The QQ plot is a graphical technique for diagnosing differences between distributions; the normal theoretical quantiles are plotted on the x-axis and quantiles found in the data on the y-axis. If the sample data is normally distributed it should form a straight line, the gradient of which is equal to the standard deviation of the sample and the value at $x = 0$ the mean. Figures 5.10 and 5.11 show cases where all the data appears to be normally distributed. Data of this sort provides little scope for further refinement of the protocol but does reinforce the belief that the ideal value is being approached.

QQ plots also provide an excellent indication of the threshold values, above or below which, bond length differences appear to be no longer behaving normally (see figure 5.12). QQ plots for all bond types with more than 100 instances in the dataset are given below. The order of the bond between the atoms is irrelevant in this analysis because the *difference* in bond length is being considered — this is beneficial as it allows for alternative bonding patterns being applied.

Outliers can also be identified by plotting a basic x-y graph of the bond length calculated by MOPAC against the bond length calculated by GAMESS. However this becomes more difficult with larger datasets. Figure 5.13 shows the C–C bond lengths in this form, one outlier is plain but the unusual behaviour evident in figure 5.12 is no longer clear. A combination of the two methods is therefore used; once threshold values have been determined the SVG graphing program described in section 1.12.1 was used to examine those molecules giving rise to the outliers.

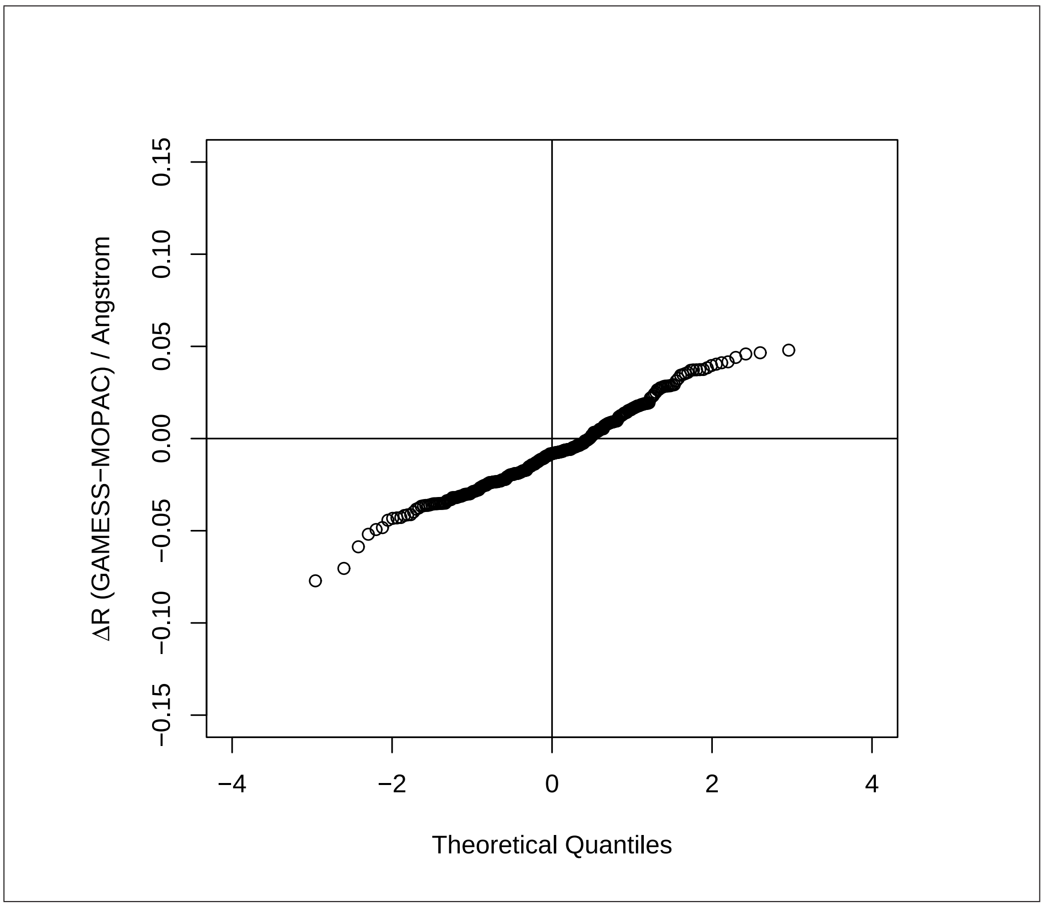


Figure 5.11: QQ plot for the 324 N–N bonds in the dataset — the data are normal over the entire length with $\overline{\Delta R}$=−0.007Å and s =0.022Å.

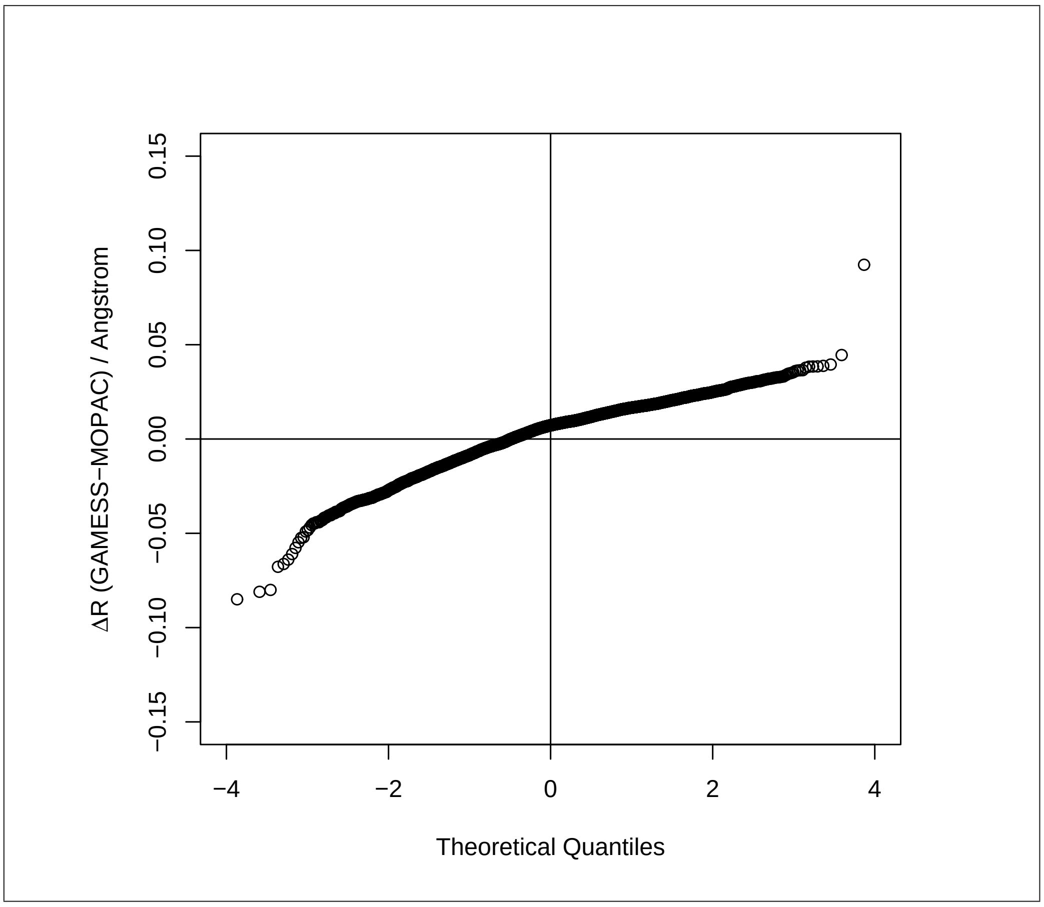


Figure 5.12: QQ plot for the 9115 C–C bonds in the dataset — the data are normally distributed within $\Delta R = \pm 0.05$Å with $\overline{\Delta R}$=0.005Å and s=0.013Å. Outliers are apparent at $\Delta R \approx 0.1$Å and $\Delta R < -0.05$Å.

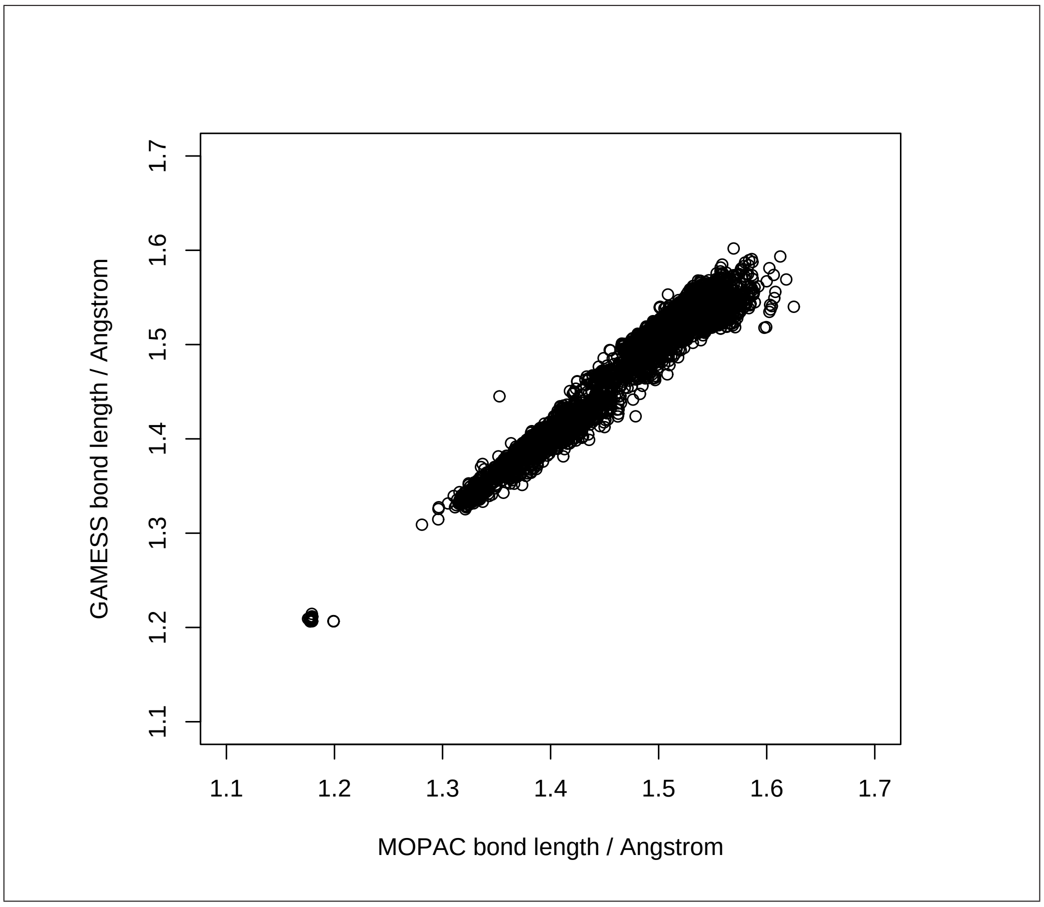


Figure 5.13: The x-y plot of all the C–C bonds shows a clear outlier but little further immediate information.

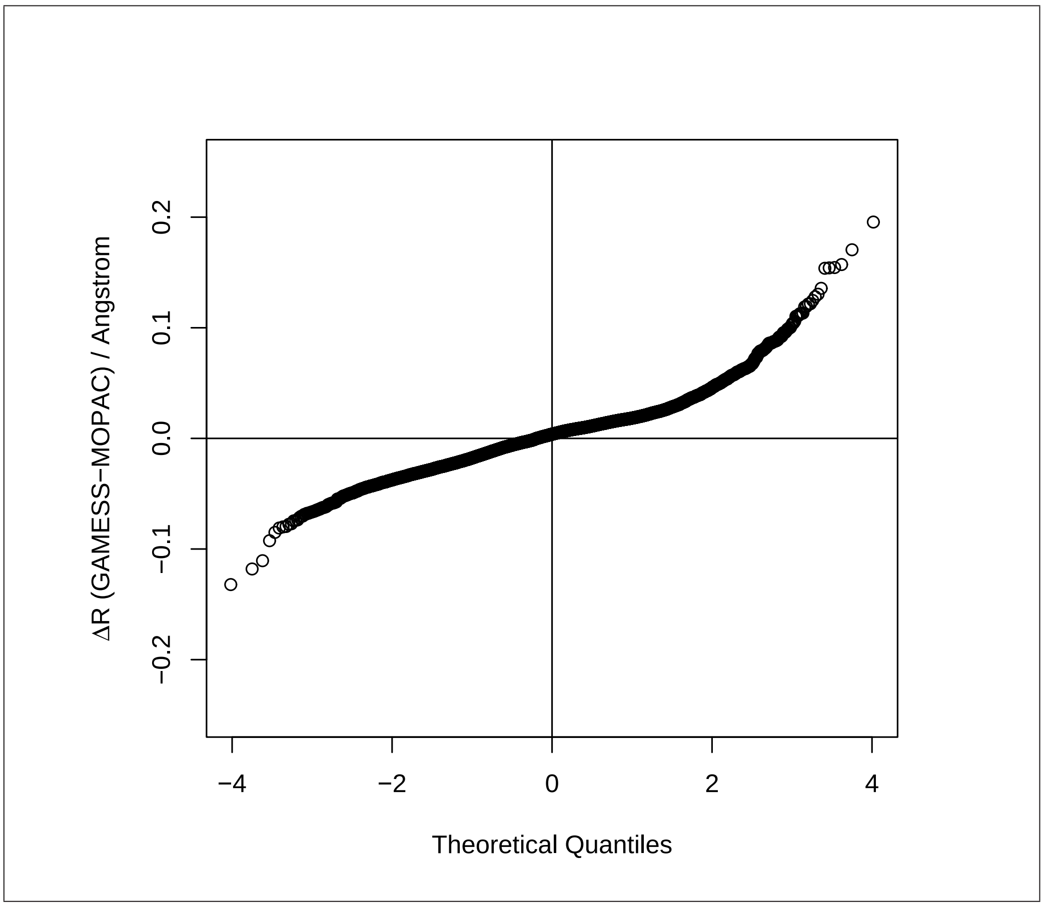


Figure 5.14: QQ plot for all the bonds in the dataset (16978) — the data are very approximately normally distributed within $\Delta R = \pm 0.1$Å with $\overline{\Delta R}$=0.002Å and s=0.020Å.

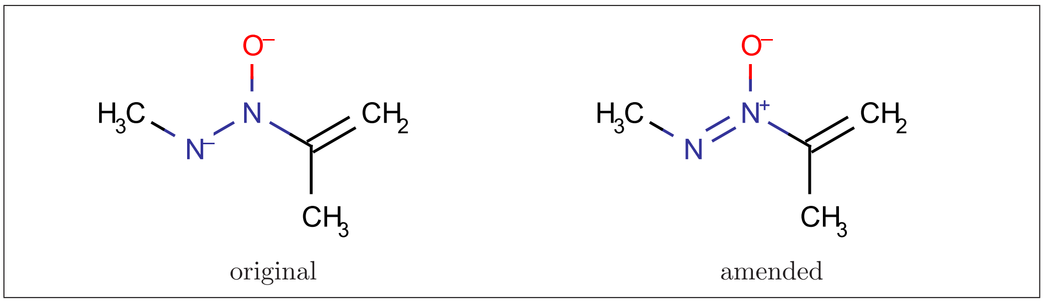


Figure 5.15: The molecule giving rise to the major outlier in figure 5.12 (left) and the emended entry now in the NCI database.

5.4.1 All Bonds

Figure 5.14 shows that overall there is good agreement between the bond lengths calculated by MOPAC and by GAMESS. Outliers are present ($|\Delta R| > 0.1$Å) but these are usually caused by the presence of heavier elements (aluminium, silicon, phosphorus and sulfur). While this graph gives an overall picture, analysis based on specific bond types is preferable.

5.4.2 C–C bonds

Figure 5.12 shows the QQ plot for all C–C bonds. The major outlier $\Delta R \approx 0.1$Å was caused by the molecule in figure 5.15. The charge and connection table for this molecule were incorrectly specified in the NCI dataset, although it has now been amended. The bonds giving rise to the tail of the distribution ($\Delta R < -0.05$Å) were found to be atoms bonded to fluorine. The 29 molecules with a C–CF$_n$ fragment ($n = 1, 2, 3$) were examined; the ΔR(GAMESS$-$MOPAC) of the C–CF$_n$ bond ranged from -0.003Å to -0.081Å. There appears to be a correlation between the number of fluorine atoms and the change in bond length $\rho = -0.75$ (see figure 5.16).

5.4.3 C–N bonds

The C–N bonds show generally good normality (figure 5.17); of those below the threshold value of $\Delta R < -0.075$Å one was from the molecule in figure 5.15, four from N-hydroxy-amide derivatives and two from molecules

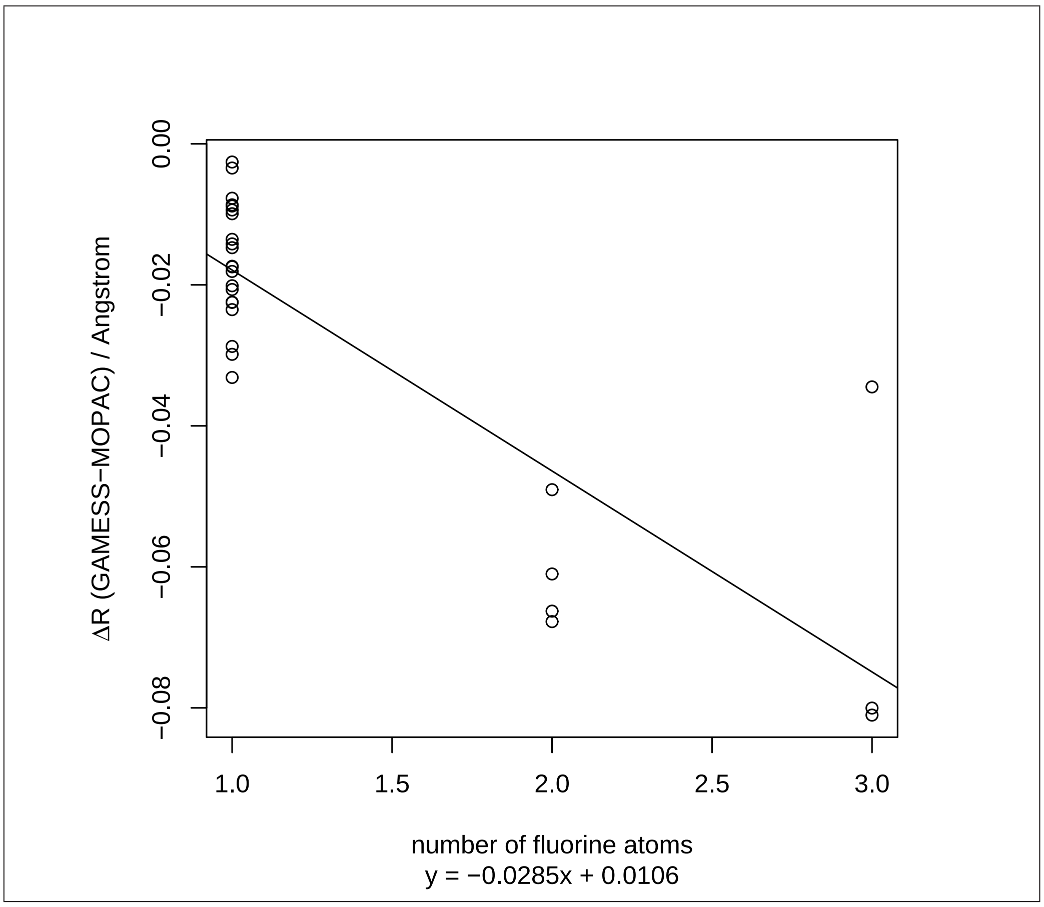


Figure 5.16: The variation in ΔR(GAMESS−MOPAC) for the 29 molecules containing a C–CF$_n$ $n = 1, 2, 3$ fragment.

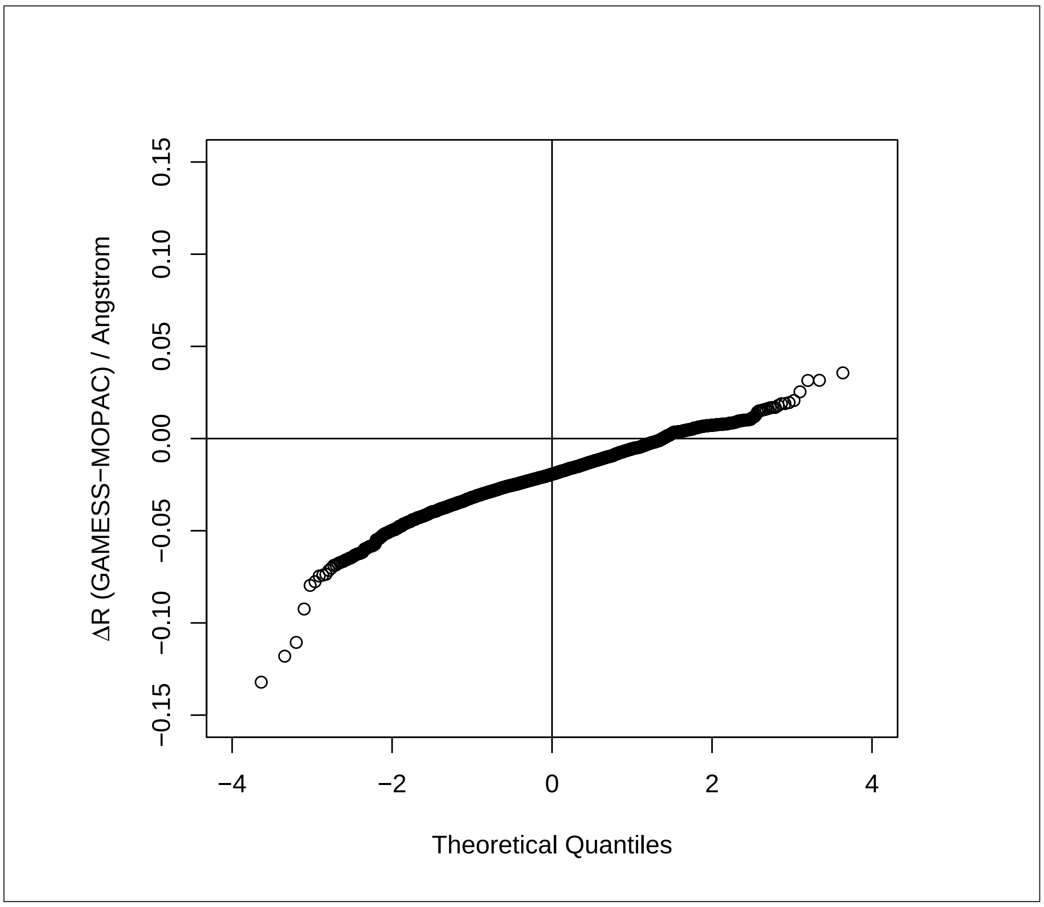


Figure 5.17: QQ plot for the 3615 C–N bonds in the dataset — the data are normally distributed for $\Delta R > -0.075$Å with $\overline{\Delta R}$=-0.019Å and s=0.014Å.

containing N–N bonds. 18 instances of N-hydroxy-amides were found in the corpus with the ΔR(GAMESS$-$MOPAC) of the nitrogen–carbonyl carbon bond varying between -0.03Å and -0.12Å. The consistent negative sign of ΔR($-$s)uggests that the discrepancies in the bond lengths are caused by systematic problems with one or both of the methods used.

5.4.4 C–O and N–O bonds

The QQ plots of C–O and N–O bond length changes (figures 5.18 and 5.19) both show kinks indicating that there might be overlapping distributions present. This is evident in the density plots in figure 5.20. An analysis to determine the molecular features that discriminate between molecules giving inconsistent bond lengths between MOPAC and GAMESS was therefore undertaken.

A dataset of 282 molecules which contained a bond with a large bond length deviations[§] (the *bad* set) and a dataset of 2302 molecules which did not (the *good* set) were prepared and 146 MOE descriptors [165] were created for the molecules in the two sets. A correlation analysis indicating which variables are likely to contribute to an agreement between the two programs and those likely to contribute to disagreement was performed. This showed that higher carbon valence connectivity, larger hydrophobic Van der Waals surface area, larger numbers of carbon atoms, higher total negative partial charge and a higher number and surface area of hydrogen atoms correlate with the *good* set. Conversely, molecules in the bad set typically contained a larger number of nitrogen atoms, larger fractional polar surface area, a large total polar surface area and high molecular mass density. Overall, many of the variables are related implicitly to the nitrogen content of the molecules (a property which was explicitly observed).

In order to establish which structural features are responsible for disagreement between the two datasets, circular fingerprints (the representation of

[§]A large bond length deviation was taken to be a bond with $|\Delta R| \geqslant$ threshold value for that bond type.

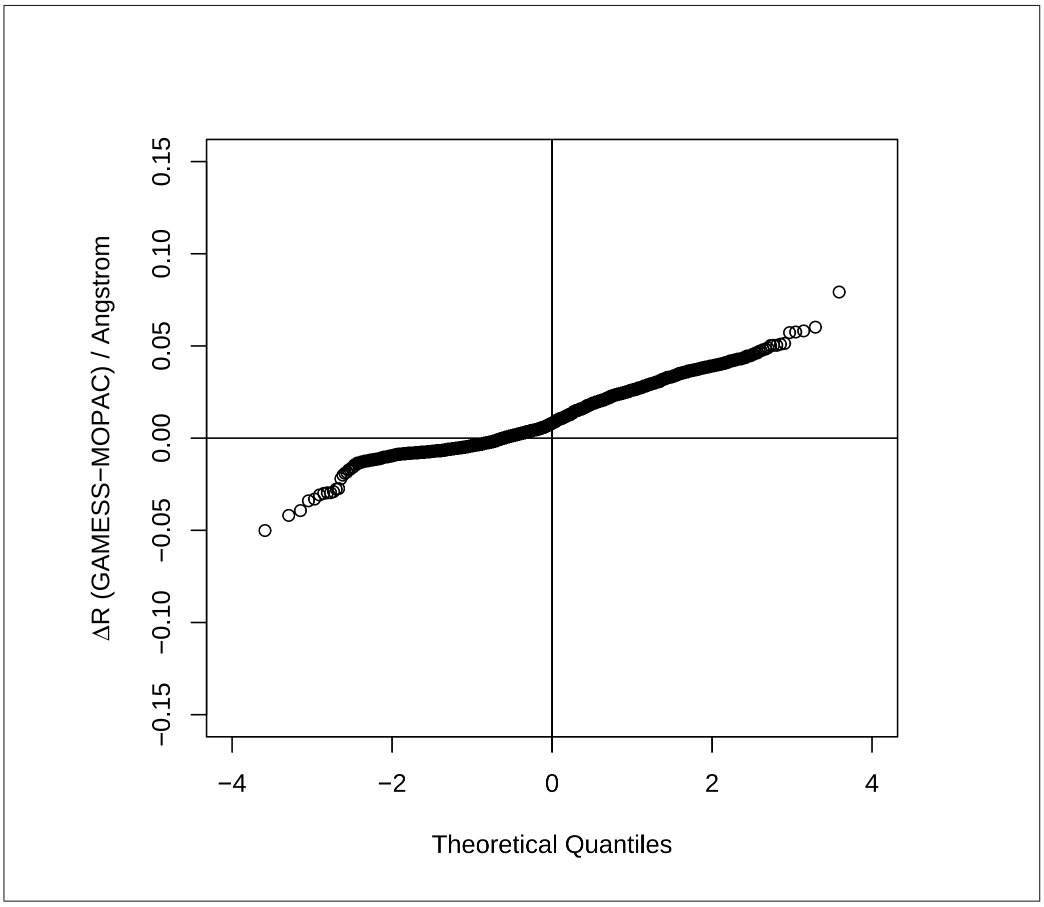


Figure 5.18: QQ plot for the 3019 C–O bonds in the dataset — the data are approximately normally distributed over the entire length with $\overline{\Delta R}$=0.011Å and s=0.014Å, although there is a kink which indicates the possibility of overlapping distributions (see figure 5.20).

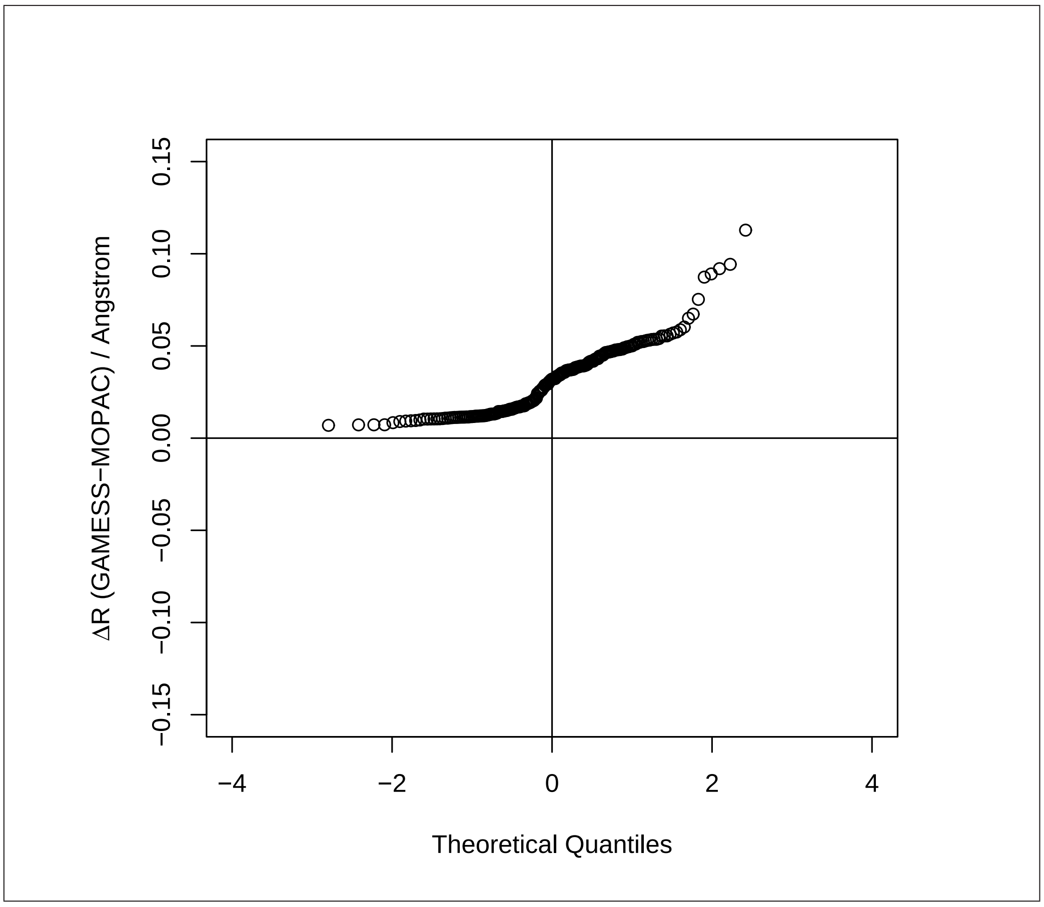


Figure 5.19: QQ plot for the 193 N–O bonds in the dataset — the data does not appear to be normally distributed, possibly there are two or more overlapping distributions present (see figure 5.20), overall $\overline{\Delta R}$=0.033Å and s=0.023Å.

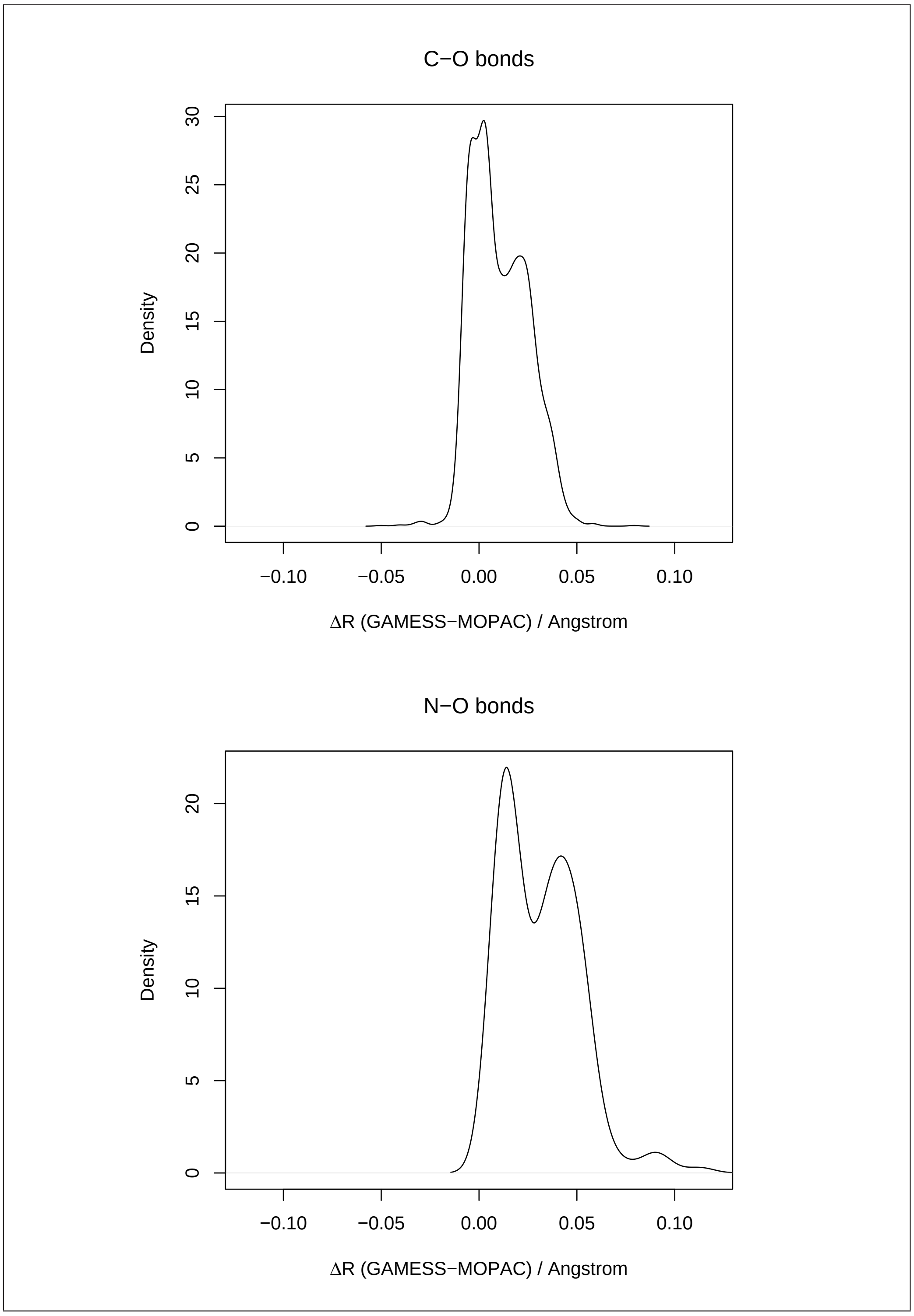


Figure 5.20: The C–O bonds appear to come from two distinct distributions and the N–O bonds from three.

molecular structures by atom neighborhoods) were employed in combination with information-gain feature selection [166]. The ten features most able to discriminate between the datasets, their relative frequencies in the datasets and the corresponding information gain are given in figure 5.21. The fragments selected are in good agreement with those determined via decision trees based on the MOE descriptors. Aromatic moieties are much more frequent in the *bad* dataset, in particular those containing nitrogen.

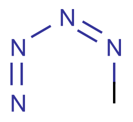
The identification of discrepancies in the bond lengths and the subsequent elucidation of the underlying cause being (nitrogen containing) aromatic moieties was performed by methods which could be entirely automated. Optimising the geometry of nitrogen, especially when the lone pair is delocalised, has been found to be problematic in the past [167]. The results presented above therefore reinforce the idea that machines can be used to validate results and furthermore that there may be undiscovered science in the legacy literature.

5.4.5 C–S bonds

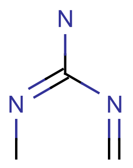
The most apparent outlier ($\Delta R \approx -0.05\text{Å}$) was caused by trifluoromethanthiol. It was thought that this might be related to the effect seen for fluorinated carbon bond lengths but the corpus only contained one molecule with a S–C–F fragment, therefore no further analysis was possible. The remaining outliers arose from bonds in aromatic heterocycles containing N–N bonds.

5.5 Time

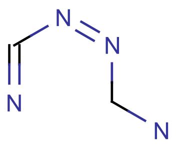
The prediction of calculation times and likely failure rates are vital for the integration of a program into a workflow and for the development of a suitable protocol. The run times of the GAMESS jobs has been examined and models of the data proposed. One of the initial constraints imposed in future protocols (that molecules containing more that 15 non-hydrogen atoms are not suitable for calculation) is a direct result of this work (see chapter 7) and the performance of the models is reported in section 8.6.



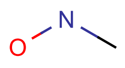
$F_b = 0.067,\ F_g = 0.000,\ I_g = 0.024$



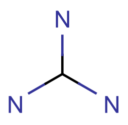
$F_b = 0.089,\ F_g = 0.004,\ I_g = 0.020$



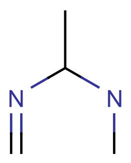
$F_b = 0.057,\ F_g = 0.000,\ I_g = 0.024$



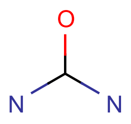
$F_b = 0.057,\ F_g = 0.004,\ I_g = 0.020$



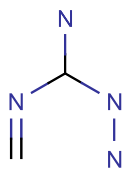
$F_b = 0.074,\ F_g = 0.004,\ I_g = 0.017$



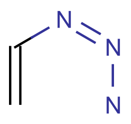
$F_b = 0.043,\ F_g = 0.000,\ I_g = 0.013$



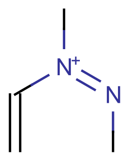
$F_b = 0.103,\ F_g = 0.017,\ I_g = 0.013$



$F_b = 0.035,\ F_g = 0.000,\ I_g = 0.012$



$F_b = 0.053,\ F_g = 0.003,\ I_g = 0.012$



$F_b = 0.043,\ F_g = 0.001,\ I_g = 0.012$

Figure 5.21: The ten molecular fragments most able to discriminate between the good and bad datasets. F_b is frequency of the fragment in the bad dataset, F_g the frequency in the good dataset and I_g is the information gain.

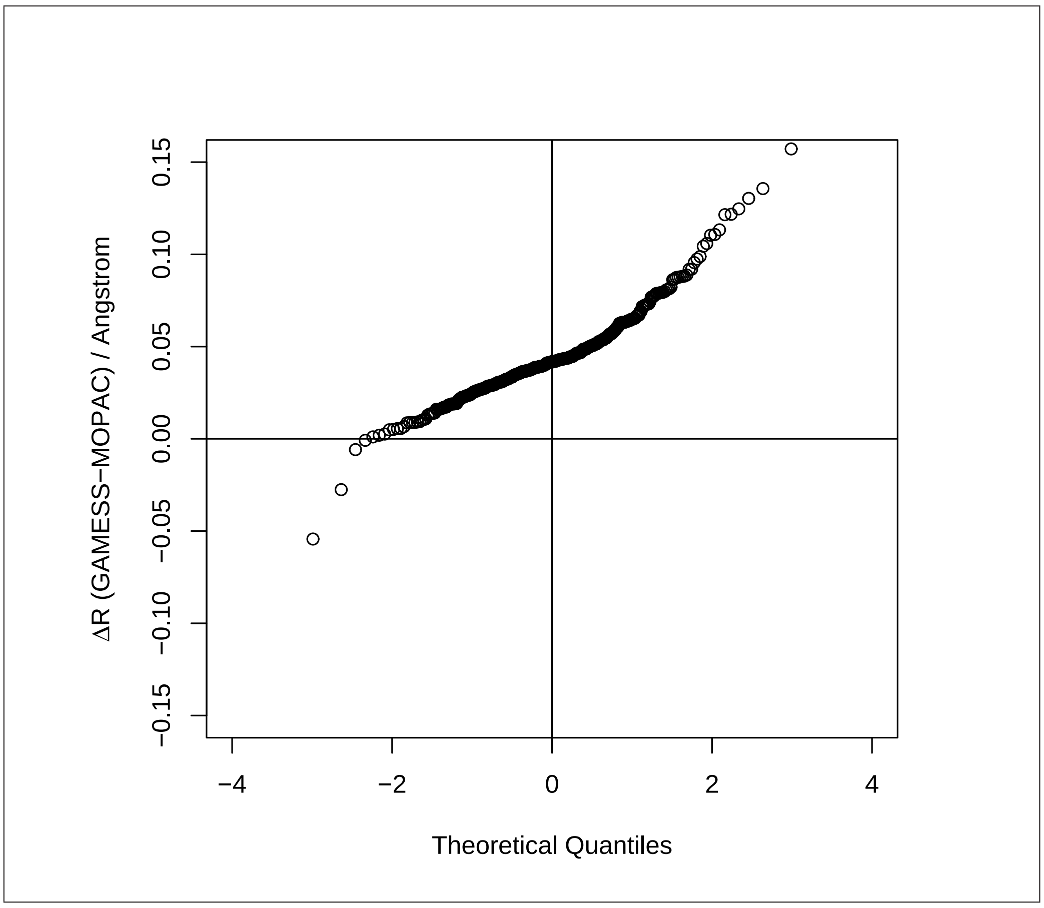


Figure 5.22: QQ plot for the 358 C–S bonds in the dataset — the data appears to be approximately normally distributed for $\Delta R > 0$Å with $\overline{\Delta R}$=0.045Å and s=0.025Å. There might be two overlapping distributions present (the behaviour changing at $\Delta R \approx 0.075$Å). However, the decreasing density of points for $\Delta R > 0.1$Å suggests that these are outliers rather than a different distribution.

number of non-H atoms	total jobs	number of failures	failure %
3	2	0	0
4	23	1	4.3
5	125	3	2.4
6	93	3	3.2
7	448	5	1.1
8	6	0	0
overall	697	12	1.7

Table 5.2: The number of calculations that failed to complete because of insufficient time, broken down by the number of non-hydrogen atoms. The statistics are based on the 685 completed jobs reported in table 5.1 and the 12 failed jobs created producing that data.

The percentage of jobs that fail to complete because they ran out of time was found to be approximately 2% (see table 5.2) but this was expected to increase with increasing molecule size. A greater proportion of the calculations on such molecules would be expected to run for the total specified run time (a week) with the effect of increasing the reported total average time. The analysis of the run times below is based only on those jobs that successfully completed the geometry optimisation within the time limit specified.

The time required for a single step of a calculation is predicted by the DFT equations to rise with the number of basis functions to the fourth power . However these algorithms have been rewritten with efficiency in mind and it is found that it is often the matrix manipulations which dominate (which scale as the cube on the number of basis functions). Figure 5.23 shows the behaviour of the mean step time for the calculations in this study which scale more favourably than the expected behaviour. It is likely to have been artificially lowered because of the removal of the all non-terminating calculations — although pseudo-diagonalisation of the matrices and symmetry effects are also likely to have contributed.

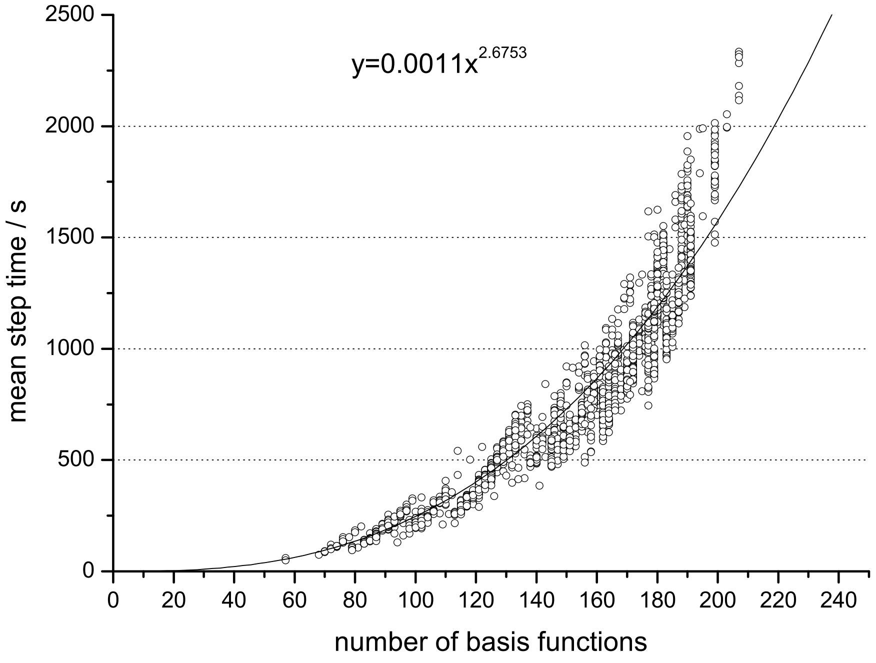


Figure 5.23: The mean step time for a calculation in this study scales more favourably than the expected n^3.

It is more convenient to work with the number of atoms in a molecule than the number of basis functions because it is readily apparent and more generalisable being independent of the basis set used. Fortunately the number of basis functions is correlated with the number of atoms ($\rho = 0.81$) but it is more strongly correlated ($\rho = 0.95$) with the number of non-hydrogen atoms (see figure 5.24).

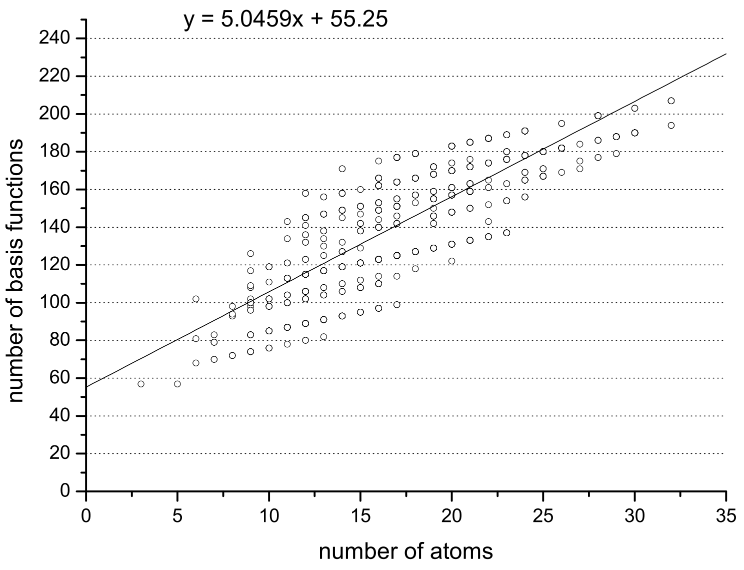
The total time to complete a geometry optimisation is not accurately predictable by theory because of the chaotic behaviour displayed near the minimum on the potential energy surface[¶]. However the total time does show a correlation with the number of non-hydrogen atoms (see figure 5.25); the calculated exponent was felt to be sufficiently close to three that a cubic dependance could be used to model the data. The use of a cubic exponent (or possibly higher) rather than the value of 2.9469 shown in figure 5.23 was also justified by recalling that the longest running jobs have been removed from the data and is the expected exponent for DFT calculations implementing an auxiliary basis rather than a standard coulomb integral. The equation of the line of best fit was found to be;

$$\sqrt[3]{t} = 3.095n + 1.5268 \tag{5.1}$$

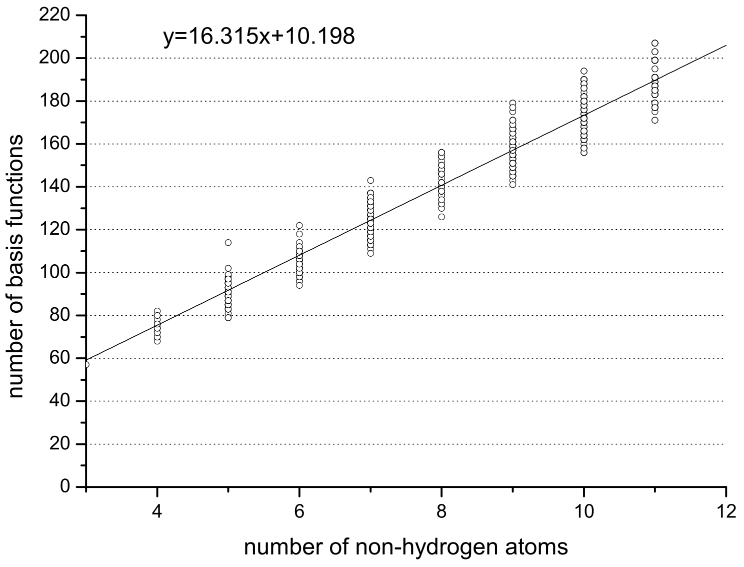
where t is the expected calculation time in seconds and n is the number of non-hydrogen atoms.

There is a variation of several orders of magnitude in the calculation time per non-hydrogen atom which results in large standard deviations. It should be noted that the standard deviation is not constant but in fact appears to rise linearly with the mean time per non-hydrogen atom as shown in figure 5.26. This is advantageous because in combination with equation 5.1 it allows the prediction of the standard deviation for larger molecules as shown in table 5.3. The ability to predict run times (with some measure of the spread of

[¶]Close to minima on potential energy surfaces the gradient with respect to all coordinates tends to zero. This often causes gradient-following algorithms to oscillate around the true minimum for an unpredictable number of steps.



$\rho = 0.81$



$\rho = 0.95$

Figure 5.24: The number of basis functions can be more accurately modelled as a function of non-hydrogen atoms than of the total number of atoms.

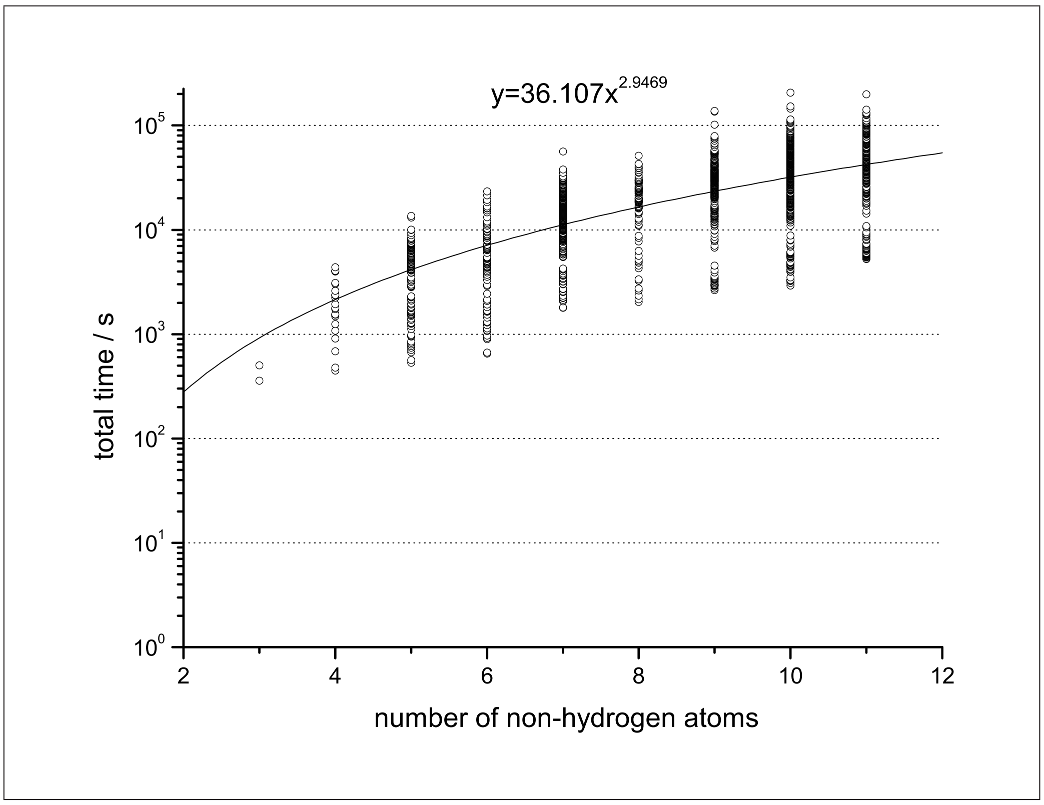


Figure 5.25: The total calculation time appears to be dependent on the number of non-hydrogen atoms present ($\rho = 0.66$) although the run times for a particular number of non-hydrogen atoms varies by several orders of magnitude.

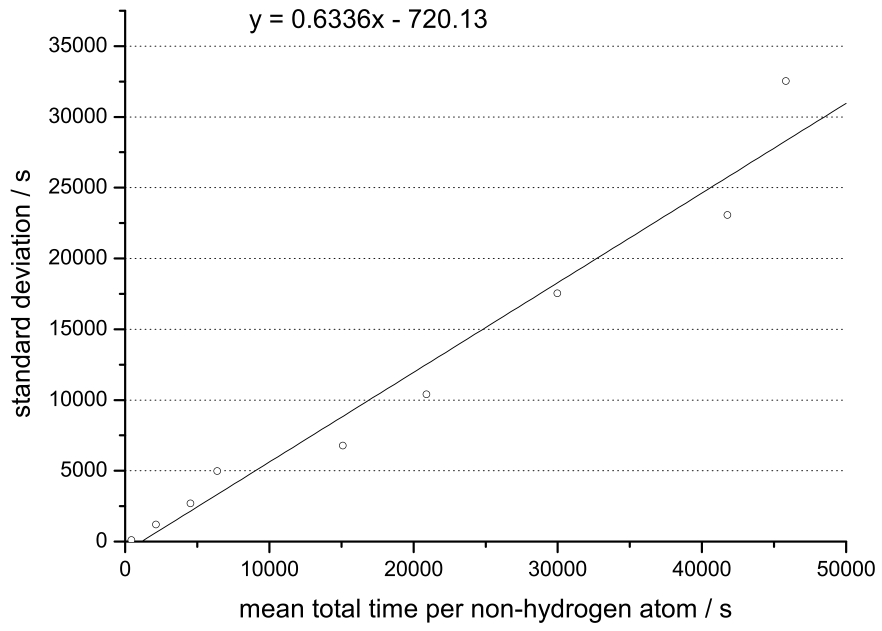


Figure 5.26: The standard deviation of run times of jobs containing a specified number of non-hydrogen atoms can be modelled by the mean run time for those jobs ($\rho = 0.98$).

these times) allows suitable protocols which may form part of a workflow to be created.

5.6 Conclusions

The MOPAC–GAMESS comparisons show good agreement in general and have shown that machines are capable of reproducing the results found by humans. However, only comparing two computational methods on data of inconsistent and indeterminable quality, means that it is impossible to determine which is *right* when a consistent discrepancy is found between the two sets of results (for example the fluorinated compounds). It then becomes necessary to validate the calculated results with those that are experimentally determined and of measurable quality (see chapter 7).

The analysis of the run times have shown that they appear to be accurately modelled using simple cubic relationships. It does not appear to be possible to predict the run time for an individual molecule — the run time for a particular number of non-hydrogen atoms varies by at least an order of magnitude. However, if a sufficiently large number of calculations are run, their behaviour may be modelled by using the mean run time (for which the standard deviation also appears to be predictable).

predicted	number of non-H atoms	mean total time / s	standard deviation /s	upper bound 95% confidence
	3	400	100	1400
	4	2100	1200	4700
	5	4500	2700	9700
	6	6400	5000	17000
	7	15000	6800	27000
	8	21000	10000	40000
	9	30000	18000	56000
	10	42000	23000	76000
	11	45000	33000	100000
✓	12	58000	36000	130000
✓	13	73000	45000	160000
✓	14	90000	56000	200000
✓	15	110000	69000	250000
✓	16	130000	84000	300000
✓	17	160000	100000	360000
✓	18	190000	120000	420000
✓	19	220000	140000	500000
✓	20	260000	160000	580000
✓	21	290000	190000	670000
	all values given to 2 significant figures			

Table 5.3: The mean run time for a calculation involving a specified number of non-hydrogen atoms may be predicted using equation 5.1 and the standard deviation calculated from this value. The 95% confidence interval is calculated by $\bar{t} + 2s$ where $\bar{t}$ is the mean runtime per non-hydrogen atom and s is the standard deviation of the sample. There are 604800 seconds in a week, thus ca. 95% of calculations involving 20 non-hydrogen atoms would be expected to complete in this time.

Chapter 6

X-Ray Crystallography

X-ray crystallography uses the diffraction pattern produced by bombarding a single crystal with X-rays to solve the crystal structure [168]. The diffraction pattern is recorded and then analysed or *solved* to reveal the nature of the crystal. This technique is widely used in chemistry and biochemistry to determine the structures of an immense variety of molecules. When single crystals are not available, related techniques such as powder diffraction or thin film X-ray diffraction coupled with lattice refinement algorithms such as Rietveld refinement [169] may be used to extract similar, though less complete, information about the nature of the crystal.

The crystallographic field also suffers from data deluge. Figure 6.1 shows the trend in the number of crystal structures published per year [170]. Currently only ca. 20% of the crystal structures determined are actually published; this figure is likely to rise in the future witch will further exacerbate the problems associated with the data deluge. These problems may be alleviated if the data could be more easily handled by automated processes, which would also allow it to be searched more precisely and easily.

6.1 Determining the Structure

The spacing in the crystal lattice can be determined using Bragg's law [171]:

$$n\lambda = 2d \sin\theta \tag{6.1}$$

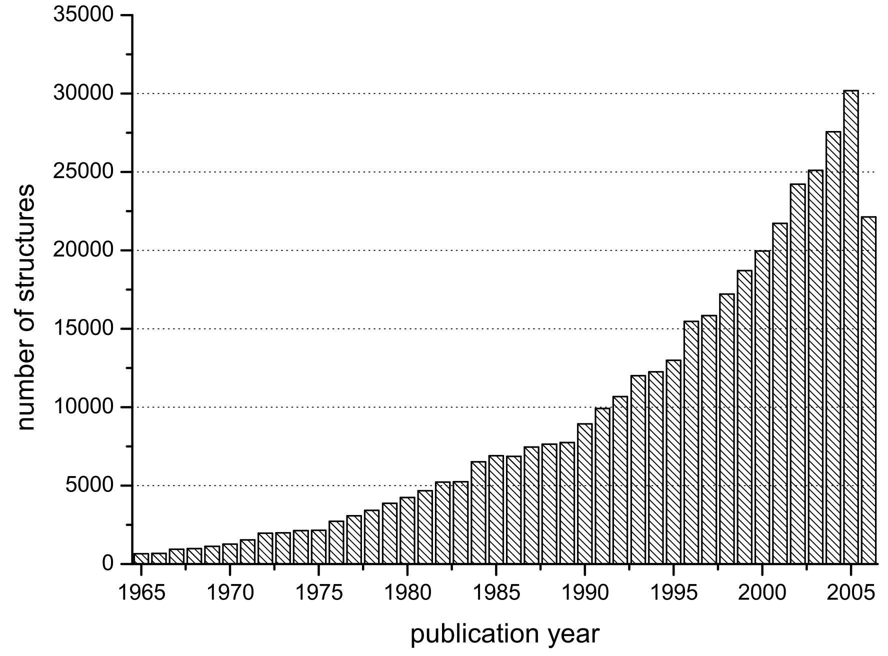


Figure 6.1: Number of crystal structures published in each of the years 1965 – 2006; the data for 2006 are as yet incomplete.

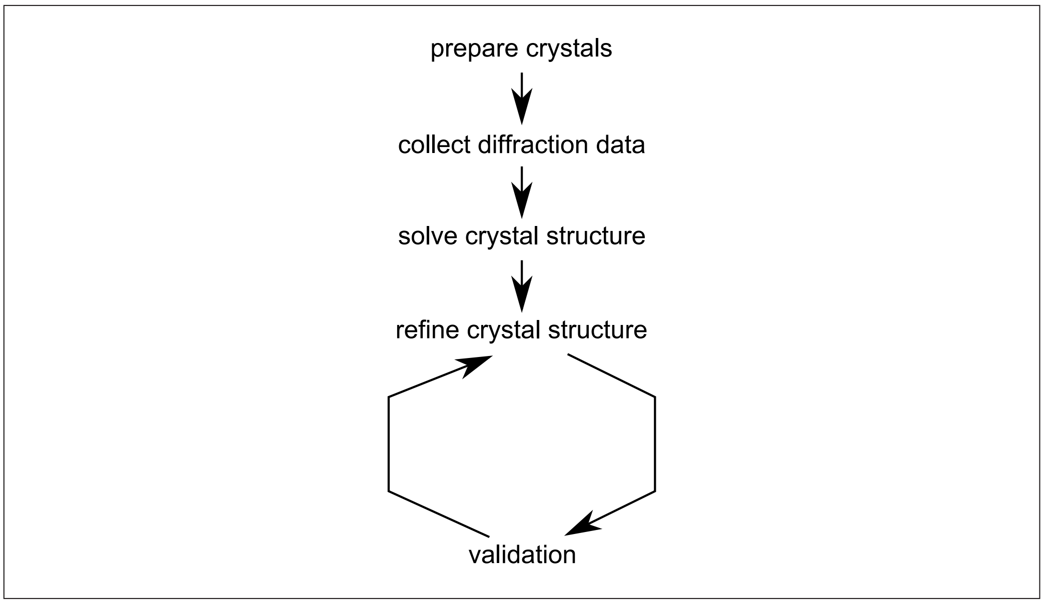


Figure 6.2: The steps involved in a crystal structure determination.

where λ is the wavelength of the incident radiation, θ is the angle of inference, d is the distance between atomic layers and n is an integer. The electrons that surround the atoms, rather than the atomic nuclei themselves, are the entities that physically interact with the incoming X-ray photons. The contribution to the diffracted X-ray intensity is larger for atoms with a greater electron density than for the lighter elements (especially hydrogen) which can make it difficult to determine the positions of the lighter elements accurately when heavy elements are present. Figure 6.2 shows the flowchart for crystal structure determination [172].

In order to solve the three-dimensional structure of a molecule, it must first be crystallized. This is because a single molecule in solution has insufficient scattering power alone, and the scattering of multiple molecules in a concentrated solution is too convoluted to yield high resolution information (although methods such as small angle X-ray scattering can be used for the determination of particle systems in terms of averaged particles sizes or shapes in such cases). A crystal can be considered to be an (effectively) infinite repeating array of the molecule of interest. The Laue conditions and Bragg's law show that constructive interference between diffracted X-rays that are in-phase reinforce each other, so that the diffraction pattern becomes detectable [173]. The geometric conditions where diffraction occurs can be visualised using Ewald's sphere [174].

Once prepared, the crystals are harvested and then *mounted*. Several methods of mounting exist: it is possible to hold the crystal in a thin glass tube using grease or by using superglue or epoxy resin to hold the crystal to a glass fibre. A more recent alternative is to use a drop of oil and liquid nitrogen to fix the crystal to the fibre. By cooling crystals the radiation damage incurred during data collection is reduced and thermal motion within the crystal decreased, giving rise to better diffraction limits and higher quality data.

Crystals are then mounted on a diffractometer coupled with a machine that emits a beam of X-rays. The X-rays are diffracted by their interaction with the electrons in the crystal, and the pattern of diffraction is recorded on film or, more recently, charge-coupled device detectors and scanned into a computer. Successive images are recorded as a crystal is rotated within the X-ray beam.

Before the advent of cryocooling, data was usually collected at room temperature. Increased radiation damage to the crystal meant that sometimes several crystals had to be used to obtain a single dataset. Cryocooling has reduced this problem. Moreover, instead of collecting the data spots one at a time, many modern machines use an array of X-ray detectors to collect data over a large range of angles at once.

The data collected from a diffraction experiment is a reciprocal space representation of the crystal lattice. The position of each diffraction *spot* is governed by the size and shape of the unit cell, and the inherent symmetry within the crystal. The intensity of each diffraction spot is recorded, and is proportional to the square of the structure factor amplitude. The structure factor is a complex number containing information relating to both the amplitude and phase of a wave. In order to obtain an interpretable electron density map, phase estimates must be obtained (an electron density map allows a crystallographer to build a starting model of the molecule). This is known as the phase problem, and can be solved in a variety of ways:

- molecular replacement — if a structure exists of a related protein, it can be used as a search model in molecular replacement to determine the orientation and position of the molecules within the unit cell. The phases obtained this way can be used to generate electron density maps.
- heavy atom methods — if high molecular weight atoms (not usually found in proteins) can be soaked into the crystal, direct methods or Patterson-space methods can be used to determine their location and to obtain initial phase estimates.
- *ab initio* phasing — if high resolution data exists (with accuracy better than 1.6Å) direct methods can be used to obtain phase information.

Having obtained initial phases, an initial model (the hypothesis) can be built. The Cartesian coordinates of atoms and their respective Debye-Waller factors (accounting for the thermal motion of the atom) can then be refined to best fit the observed diffraction data. This generates a new (and hopefully more accurate) set of phases and a new electron density map is generated. The model is then revised and updated by the crystallographer and a further round of refinement is carried out. This continues until the correlation between the diffraction data and the model is maximised.

6.2 Derived Results

The *primary* numerical results of a structure determination are the parameters obtained in the least-squares refinement. Usually these consist of:

- three positional coordinates for each atom

- and a number (often one, for isotropic, or six, for anisotropic models) of *temperature factors*, *thermal parameters* or *displacement parameters* for each atom
- other parameters for effects such as extinction and overall scaling of the observed and calculated datasets

These parameters may be refined as supposedly independent values, or there may be various constraints and/or restraints applied to their refinement. The refinement supplies not only values for the parameters but also an estimated standard deviation (esd) for each one. The structure determination also gives the atom types.

The primary results are usually not the main object of the structure determination experiment, rather, it is the molecular geometry and possibly intermolecular interactions which are of interest. Therefore secondary results must be derived (such as bond lengths, bond angles and torsion angles). Each of these derived values should also have an associated esd.

It has been suggested that the thermal parameters describe both the time-averaged temperature-dependent movement of the atoms about their mean equilibrium positions and their random distribution over different sets of equilibrium positions from one unit cell to another. These parameters may therefore be called atomic displacement parameters. The interpretation and analysis of these atomic displacement parameters is often not undertaken.

6.2.1 β, B and U Parameters

Atomic displacements are described by a variety of different parameters, all of which are mathematically related. Thus, for an isotropic model, a single parameter is used, but this may be called B or U. These are related by

$$f'(\theta) = f(\theta)\exp\left(-B\sin^2\frac{\theta}{\lambda^2}\right) = f(\theta)\exp\left(-8\pi^2 U\sin^2\frac{\theta}{\lambda^2}\right) \tag{6.2}$$

where $f(\theta)$ is the scattering factor for a stationary atom and $f'(\theta)$ the scattering factor for the vibrating atom. B and U both have units of Å^2 and U

represents a mean-square amplitude of vibration. For an anisotropic model, six parameters are used and the exponent $\left(-B\sin^2\frac{\theta}{\lambda^2}\right)$ becomes

$$-(\beta^{11}h^2 + \beta^{22}k^2 + \beta^{33}l^2 + 2\beta^{23}kl + 2\beta^{13}hl + 2\beta^{12}hk) \qquad (6.3)$$

although other forms are also used. These parameters are often represented graphically as thermal ellipsoids. This is possible only if certain inequality relationships among the six parameters are satisfied; otherwise they are said to be non-positive definite and the corresponding ellipsoid does not have three real principal axes. Such a situation may indicate a real problem in the structural model (*e.g.* a disordered atom), or it may be due to imprecise U^{ij} parameters (high esds), in which case the anisotropic model for this atom is perhaps not justified.

The anisotropic displacement parameters are often not published in most chemical journals and their significance is difficult to assess immediately. A simpler parameter to assess atomic motions is the equivalent isotropic parameter U_{eq}. There are different definitions of U_{eq} and some appear to be inappropriate [175]. One version of U_{eq} is that corresponding to a sphere of volume equal to the ellipsoid representing, on the same probability scale, the anisotropic parameters. A commonly used definition is $U_{eq} = \frac{1}{3}$ (trace of the orthogonalised U^{ij} matrix) although its meaning is not entirely clear. The effect of temperature on U_{eq} is examined in section 8.5.

6.2.2 Libration

Thermal motion is known to produce an apparent shrinkage in molecular dimensions. Figure 6.3 shows how this shrinkage occurs for riding motion but is extensible to all libration (rotary oscillation). If a molecule has only small internal molecular vibrations compared with its movement as a whole about its mean position in the crystal structure, then it can be treated approximately as a rigid body. In this case the atomic displacements are not independent and the U^{ij} parameters of the atoms must be consistent with the overall molecular motion.

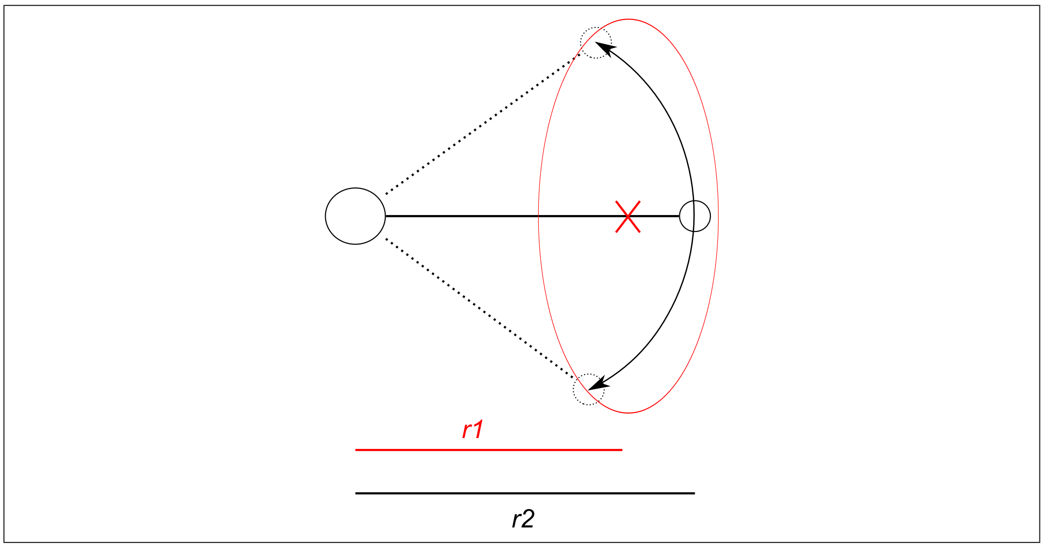


Figure 6.3: A light atom bonded to a heavy one will oscillate about its equilibrium position causing the election density to be spread out (black lines). The electron density is modelled as an ellipsoid (in the three dimensional case) with the atom at the centre of this ellipsoid (red). This causes observed bond length, $r1$, to be shorter than the true bond length $r2$.

The molecular motion can be described by a combination of three tensors (3×3 matrices): T, L and S. T describes the translation and is symmetric, L describes libration (also symmetric) and S describes a screw motion which is not symmetric. T and L both have 6 independent parameters and S has 8 because there is a constraint on the three diagonal terms. From the rigid body parameters, corrections can be calculated for bond lengths within the molecule; these depend only on the libration tensor. Although many molecules are not rigid, certain groups of atoms within them may be and it is possible to treat these as rigid bodies and to calculate the subsequent bond length corrections. The Hirshfeld test [176] is used to determine whether a molecule or part of a molecule can be regarded as a rigid body. The effect of rigid body corrections and the agreement with the calculated bond length is shown in section 8.4.5.

6.2.3 Minor Conformations and Incorrectly Assigned Atoms

It is sometimes hard to distinguish whether an atom has a high thermal motion or is statically disordered (minor conformations are present). The presence of disorder in a structure, unless it is very simple and can be well modelled, reduces to some extent the overall precision of the structure, not just the particular atoms affected. The lowest reported site occupancy in the dataset of crystallography used in this thesis was 0.0107. It is therefore possible that there were some minor conformations ($< 1\%$ frequency) which were not reported.

It is possible, though unlikely, that a completely incorrect structure is identified (essentially the wrong chemical compound is found) when the crystal structure is solved. An example of wrongly assigned atom types being reported was identified by von Schnering and Vu in 1986 [177]. They questioned the reported structure of '$[ClF_6][CuF_4]$' which in reality was probably $[Cu(H_2O)_4][SiF_6]$. The mistake was likely caused by the similar scattering powers of Si and Cl, and of O and F. Subsequent improvements in crystallography mean that such incorrect assignments are now extremely unlikely, especially when only light atoms are present in the crystal.

6.2.4 Atomic Scattering Factors

The scattering factors commonly used are reasonably accurate representations of the scattering power of individual, isolated atoms at rest. This commonly referred to as the independent atom model (IAM). The scattering factors have spherical symmetry and probably their greatest limitation is the lack of allowance for distortion of this spherical electron density when atoms are placed together and bonded to each other. This is one reason why bonds to hydrogen atoms are found to be systematically shortened in X-ray diffraction studies.

The invariom concept [178] provides a definition of a pseudoatom electron density, that is transferable between molecules, and employs the multipole formalism introduced by Hansen and Coppens [179]. In invariom refinement, the multipole parameters are predicted by a procedure involving theoretical calculations and can be described as providing aspherical scattering factors. Hence, the number of parameters to be refined in the least-squares procedure does not increase when compared with a standard IAM refinement.

In 2005 Dittrich *et al.* demonstrated that the determination of molecular geometry by conventional X-ray single-crystal diffraction experiments of organic molecules can be improved by invariom modelling [180]. The lengths of bonds involving hydrogen atoms were particularly affected. The structure of L-valinol determined by the IAM, invariom refinement and quantum chemical geometry optimisation (using *GAUSSIAN98*, D95++(3df,3pd)) was reported in 2006 [181]. Table 6.1 shows the comparison of the bond length differences between each of the methods. The effect of using spherical rather than aspherical scattering factors for atoms (other than hydrogen) is thought to be significantly less than those caused by libration, minor disorder and incorrectly assigned atoms.

X-C bonds, X=C,O,N	
	$\overline{esd}$ / Å
invariom	0.0007
IAM	0.0012
	$\overline{\Delta R}$/ Å
ΔR(invariom−IAM)	0.0010
ΔR(invariom−*GAUSSIAN98*)	−0.0034

X-H bonds, X=C,O,N	
	$\overline{esd}$ / Å
invariom	0.0089
IAM	0.0118
	$\overline{\Delta R}$/ Å
ΔR(invariom−IAM)	0.1284
ΔR(invariom−*GAUSSIAN98*)	−0.0280

Table 6.1: The effect of using invariom refinement compared to the IAM and the *GAUSSIAN98* geometry optimised structure. The values are based on data reported by Dittrich *et al.* in 2006 for L-valinol [181].

6.3 The CIF Format

As mentioned previously, X-ray crystallography, in the form of Crystallographic Information Files (CIFs), is a source of high quality, experimentally determined, 3D coordinates and connection tables for structures. The CIF format was introduced in 1991 to be used for the electronic transmission of crystallographic data between laboratories, journals and databases and was adopted by the International Union of Crystallography (IUCr) as the recommended medium for this purpose [182]. The development of a dictionary of crystallographic data items was a major feature of this work; each data item has been assigned a self-explanatory (within a 32 character limit) name for use in a CIF and the precise definition of the item given in the appendix to the paper (now available online) [183]. The CIF is based on the Self-Defining Text Archive and Retrieval (STAR) file syntax procedure defined by Hall in 1991 [184] and subsequently defined in Backus-Naur form in 1993 [185].

The CIF format places nine further restrictions on the STAR file syntax as detailed below:

1. Lines may not exceed 80 characters
2. Data names and block codes may not exceed 32 characters. All data names are case insensitive
3. The CIF dictionary defines particular data items as *number* or *character* types
4. A data item is assumed to be a *number* if it starts with a digit, plus, minus, a period and is not bounded by matching single quotes, double quotes or semicolons as the first character on a line
5. A number may be specified as an integer, floating-point number, or in scientific notation. When concatenated with a number in parentheses, that integer is assumed to be the esd in the final digit(s) or the number. For example: 34.5, 3.45E1, 34.5(12), 3.45E1(12) are all versions of 34.5 with and without an esd of 1.2

6. A data item is assumed to be of data type *text* if it extends over more than one line, *i.e.* it starts and ends with a semicolon as the first character of a line

7. A data item is assumed to be of data type *character* is if is not a *number* or *text*

8. Only one level of `loop_` is permitted. Additional levels of repeated data must be stored as lists within a text field

9. The CIF dictionary specifies default units for CIF data items. If the data item is not stored in the default units, the units code is appended to the data name. Only those units defined in the CIF dictionary are acceptable.

The data names defined for use in a CIF are separated into components representing the internal hierarchy of data categories. The data names are of the form

```
_<category>_<topic>_<subtopic>
```

For instance data relating to atoms begins with `_atom_` and is further categorised into those data relating to the position of the atom in the crystal (`_atom_site_`) and the properties of the atom type that occupies that position (`_atom_type_`).

The `_chemical_conn_` data items specify the 2D chemical structure of the molecular species and allow a 2D chemical diagram to be constructed — a complete chemical entity must be described so symmetry-generated atoms must be included. The connection tables of all the molecules in the CIF are thus trivially recoverable. However, whilst this facility is provided, it is not required and authors usually do not supply this data (no instances were found in the 6738 CIFs examined in this study).

6.4 Quality Indicators

There are many data items that can be used to assess and validate a crystallographic structure. UNIMOL [186] implements a variety of checks including searching for voids and consistency of bond lengths. The IUCr host a free service, checkCIF [187], that provides a report to authors of possible mistakes in the CIF submitted. There are three versions of checkCIF: basic structural check, full structural check and full publication check. The full structural check performs 321 tests to determine the self-consistency of the data and that it falls within expected parameters [188]. Data that do not pass a test are reported as *alerts* of differing levels (A to D, most to least severe). Whilst CIFs with level A alerts are published, the author has to justify that the reason for the alert is valid; in general authors are advised to resolve as many of the problems giving rise to alerts before submission.

The automatic checkCIF validation helps to ensure that there are no errors present in the published data (and that it is at least self-consistent) but this still allows large variation in the *quality* of the data. Fortunately the CIF contains data items that can be used as a measure of the data quality, the most obvious being the R-factor. Various methods are used to calculate the R-factor but are reported under the `_refine_ls_` data name (`_refine_` data names describe the structure refinement parameters and `_ls_` data names refer to the least squared method). The conventional R-factor is `_refine_ls_R_factor_gt` (which succeeded `_refine_ls_R_factor_obs`). This is the residual factor for reflection data classified as observed (the intensity of the reflection is sufficiently intense) and is calculated by

$$R = \frac{\sum \left| |F_m| - |F_c| \right|}{\sum |F_m|} \tag{6.4}$$

where F_m and F_c are the measured and calculated structure factors. The calculated structure for atoms with diffracting power f situated at different points in the unit cell as specified by the fractional coordinates x, y, and z

for the reflecting plane hkl is given by

$$F_c^2 = A^2 + B^2 \tag{6.5}$$

where

$$A = \sum f \cos 2\pi(hx + ky + lz) \tag{6.6}$$

$$B = \sum f \sin 2\pi(hx + ky + lz) \tag{6.7}$$

Atoms in crystals vibrate at ordinary temperatures with frequencies very much lower that those of X-rays; at any one instant, some atoms are displaced from their mean positions in one direction while those in another part of the crystal are displaced in another direction. Consequently, diffracted X-rays which would be exactly in phase if the atoms were at rest are actually not quite in phase, and the intensity of the diffracted beam is thus lower than it would be if all atoms were at rest. The thermal displacements of atoms in crystals with large plane spacings (those giving reflections at small angles) are small fractions of the plane spacing, hence the intensities are largely unaffected. However crystals with closely spaced planes are more affected. The effect also increases with rising temperature; lower temperature experiments are therefore likely to be higher quality.

Neutrons do not interact with matter in the same manner as X-rays. X-rays interact primarily with the electron cloud surrounding each atom. Neutrons interact directly with the nucleus of the atom, and the contribution to the diffracted intensity is different for each isotope; for example, hydrogen and deuterium are distinguishable. It is also often the case that light atoms contribute strongly to the diffracted intensity even in the presence of heavy atoms. Non-magnetic neutron diffraction is directly sensitive to the positions of the nuclei of the atoms. Neutron diffraction is therefore likely to provide extremely high quality structures with the positions of all the atoms well determined. Unfortunately, such experiments are not performed with the same regularity as the more traditional X-ray based studies.

Chapter 7

Creating a Workflow

A workflow may be described as a collection of steps and data that define the paths taken to complete a *task*. Such a workflow might contain *processes* such as displaying content to users, collecting information from users or computer systems, performing calculations and sending messages to external computer systems. The tasks under consideration in this chapter are those of calculating the minimised geometries for the connection tables (CTs) contained in CIFs and organising these results in a suitable way for analysis to take place.

Previously in this thesis phrases such as *the molecules were submitted for calculation* and *a protocol was created* together with workflow diagrams such as figure 5.1 have been used to indicate what processes were performed. The amount of work necessary to create these processes and to link them together is often not appreciated. This chapter examines in more detail these processes and the work necessary for their creation.

A simple high-level overview of a workflow is very simple to construct (figure 7.1 shows a typical example). Such a high-level (or coarse-grained) view is unlikely to reflect what is actually required to complete the specified task. Whilst much effort is typically put into ensuring that the calculations performed are valid and useful (these parameters are incorporated in the *protocol*, see section 7.9) it is vital that the infrastructure is in place to allow such a protocol to be adopted.

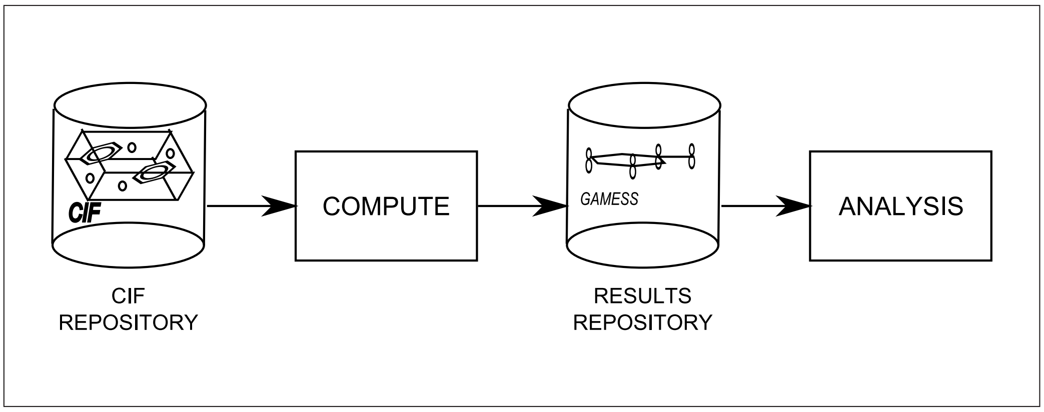


Figure 7.1: A simple high-level overview of a workflow.

7.1 Existing Workflow Technology

Murray-Rust *et al.* have examined various methods for the creation and development of workflow technology for chemistry [189, 190]. Their approach has built on the $^{\mathrm{my}}$Grid Taverna project [191] and focuses on creating *Web-Services* (WS) for each of the processes. A WS provides a standardised programmatic interface for a particular piece of functionality. This has the advantage that large libraries or platform-dependent code need not be distributed. Instead the process that they perform are encapsulated within a single WS.

A finer-grained workflow for the required tasks is shown in figure 7.2. Whilst this is more realistic than that shown in figure 7.1, none of the processes or repositories shown were immediately available in usable states. Previous experience showed that it was unrealistic to attempt to control the entire process with a single workflow; a modular approach was therefore adopted. This required the creation of lightweight and independent modules for each of the processes. The modules created would then be encapsulated in a WS and distributed for re-use. The *glue* between each of the modules is XML/CML and the JUMBO toolkit which allows the output of one activity to be the input for the next as outlined in by Zhang *et al.* in 2004 [192]. To develop the required modules (which are often entire workflows themselves)

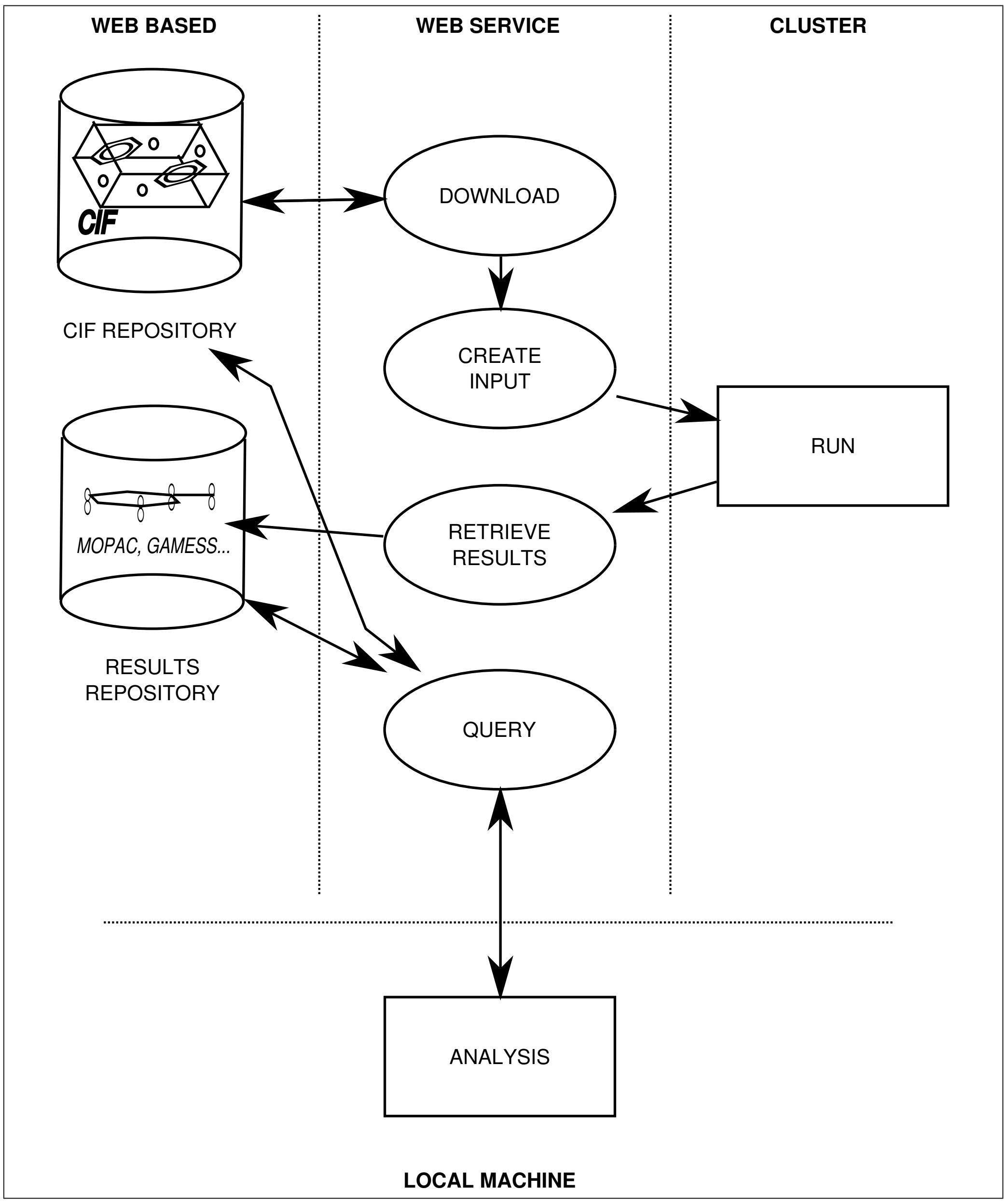


Figure 7.2: A workflow describing the ideal situation for calculating properties of a collection of CIFs and analysing the results

each of the processes and repositories must be considered at a lower level.

7.2 CIF Repositories

The Cambridge Crystallographic Data Centre (CCDC) was conceived in 1965 as a non-profit, charitable institution whose objectives are the general advancement and promotion of the science of chemistry and crystallography for the public benefit. As part of this, the CCDC compile the Cambridge Structural Database (CSD) which is the world repository of small-molecule crystal structures.

The CSD, whilst not web-based, should therefore provide an ideal repository of CIFs. Unfortunately the conditions of use of the CIFs provided from the CCDC CIF archive state that

> Individual CIF datasets are provided freely by the CCDC on the understanding that they are used for bona fide research purposes only. They may contain copyright material of the CCDC or of third parties, and may not be copied or further disseminated in any form, whether machine-readable or not, except for the purpose of generating routine backup copies on your local computer system. [193]

The intention is for all the results of this project to be openly available (and repeatable), thus it was determined that the CCDC could not be used as a CIF resource.

The electronic supplementary data provided for an article in a chemical journal may contain a CIF. This data can (often) be accessed without subscribing to the journal, is electronically searchable and can be reused. It is therefore ideal for this project except that there is no longer a single access point or access protocol.

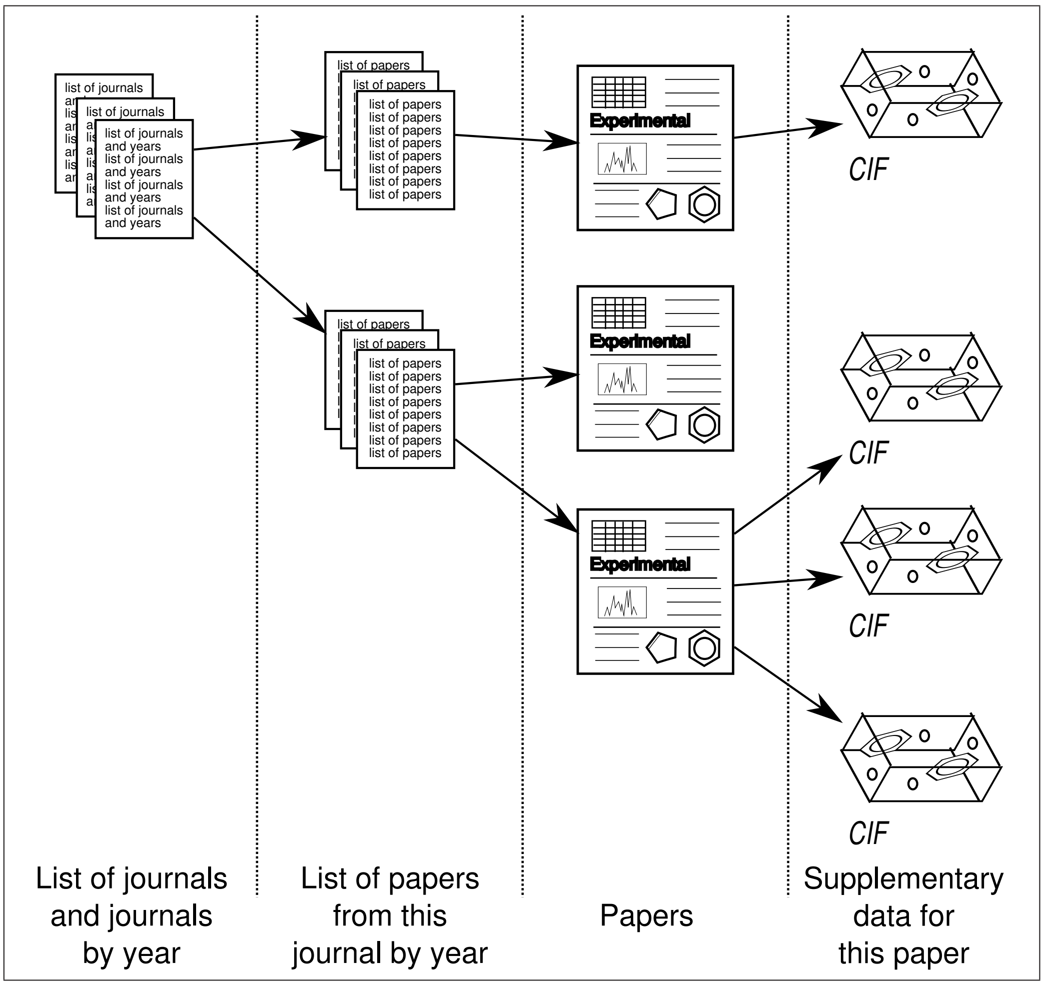

list of journals and years
list of papers
Experimental
CIF
List of journals and journals by year
List of papers from this journal by year
Papers
Supplementary data for this paper

Figure 7.3: Overview of how journals organise supplementary data.

To create a repository of CIFs for calculation it was necessary to determine how to retrieve the CIFs from a journal that would allow automated electronic download. Figure 7.3 shows how the supplementary data is typically organised by a journal. Day described a workflow for this process in 2005 [189]. This workflow was used to provide a repository of CIFs which were provided as supplementary data for articles published in Acta Crystallographica, Section E between January 2001 and September 2005. These were attractive because only a single experiment is reported in each CIF (which simplifies the parsing process) and because all the CIFs submitted to this journal must contain certain data items as specified by the IUCr [194].

7.3 Download

The download process as indicated in figure 7.2 would tend to indicate that a molecule's CT is immediately retrievable from the CIF, or at least it is possible to determine whether or not a particular CIF is useful before downloading it*. The nature of the download process thus depends largely on the availability of an online CIF repository which could be queried for structures fitting the chosen protocol. Ideally only those CIFs containing structures that will be calculated would be downloaded. Following the previous discussion this was clearly impossible. This section thus focuses on the creation of a suitable data structure to store the set of downloaded CIFs and how the CTs contained in the CIFs were made searchable.

The downloaded CIFs were provided in the structure shown in figure 7.4. Each of the CIFs is uniquely identified within the journal by a two letter, four number combination. Although unique, the identifier does not contain any information about the CIF but it is believed that each reviewer has their own two letter code and the number is simply an increment. Thus it should be possible to determine which structures were cleared for publication by which reviewer.

*In this case *useful* means that the CIF contains the CT for a structure that would be appropriate for calculation.

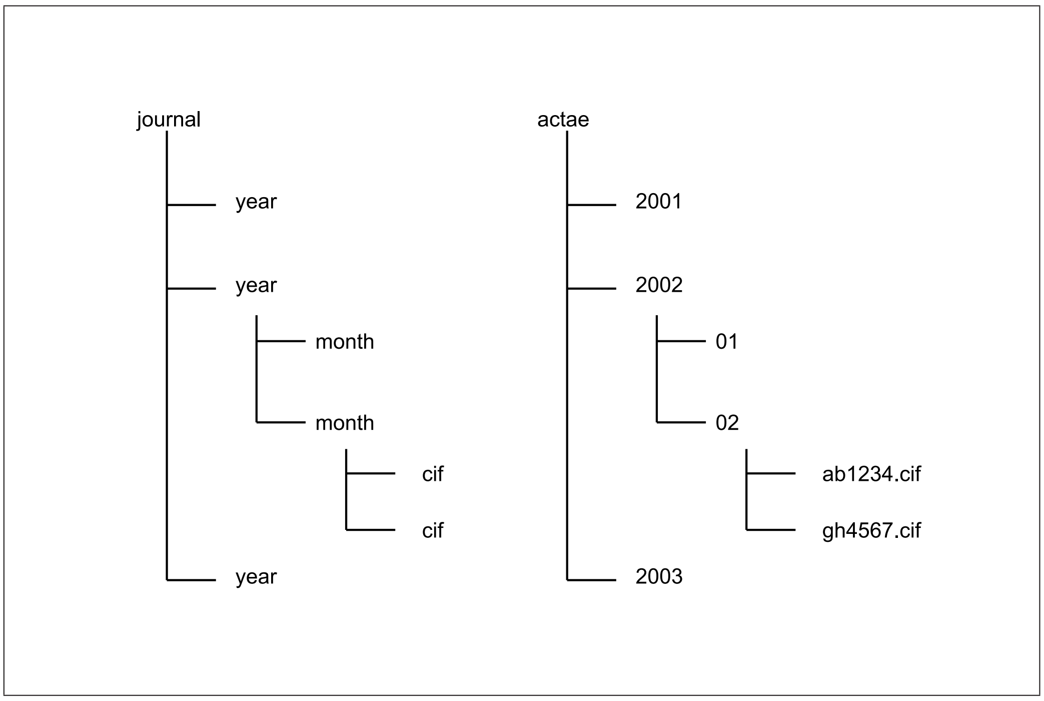


Figure 7.4: The data structure for the downloaded CIFs, the general case (left) and an example of the values for this work (right).

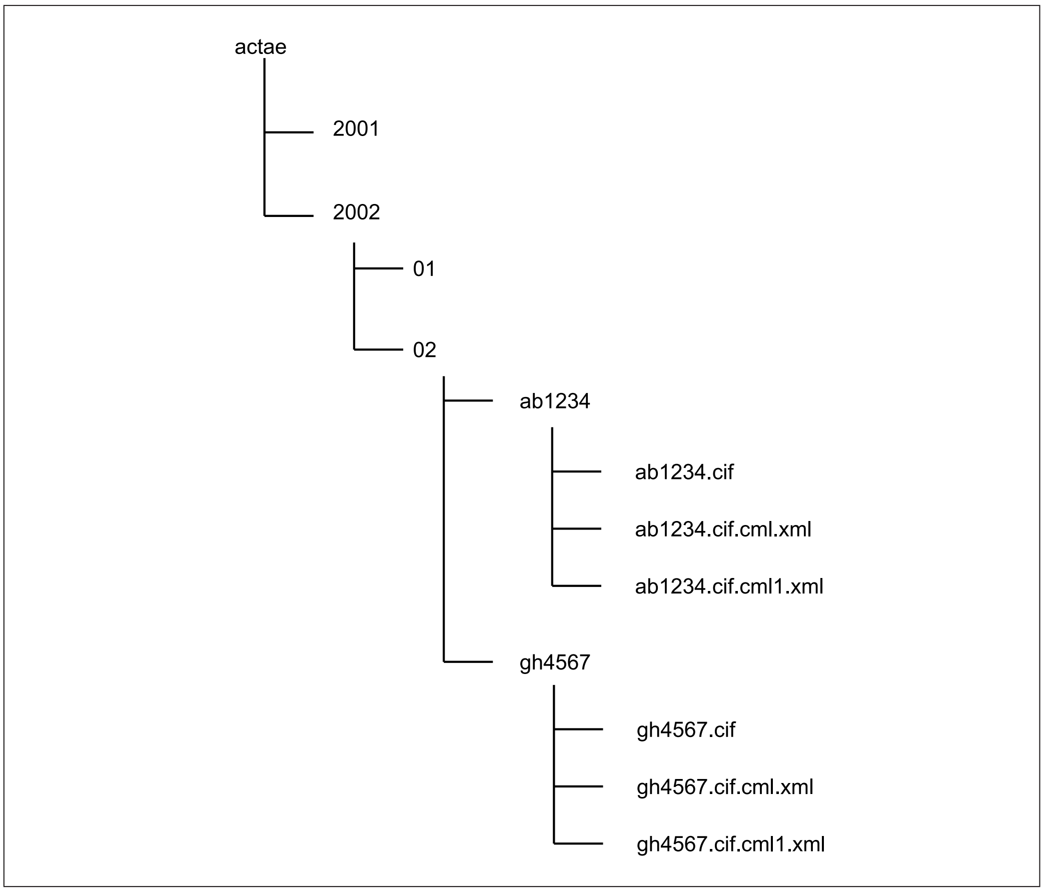


Figure 7.5: The hierarchical data structure adopted for the downloaded CIFs; the `.cif.cml.xml` contains the complete CIF parsed to CML and the `.cif.cml1.xml` contains only the molecules from the CIF for display purposes.

Although CIF is based on the STAR syntax that was designed to be machine-understandable, the CIFs must be converted to CML to be integrated into the rest of the workflow and processed with the available tools. A hierarchical structure was chosen to store the data pertaining to each CIF; thus each CIF was moved to a subdirectory of the same name. This allows all the data relating to a particular CIF to be stored under a separate subdirectory.

The CT and the 3D coordinates of the molecules in the CIF are obtained by the following processing sequence based on that published by Murray-Rust *et al.* in 2004 [195]:

1. Read the CIF into an XMLDOM [52] (achieved using a CIF2CML library [196])
2. Discard minor disordered components
3. Convert fractional coordinates to cartesian
4. Join bonds using 'reasonable' covalent radii
5. Apply symmetry operations to generate the minimum number of molecular fragments
6. Generate CT(s) for the molecule(s)
7. Check against chemical formula (often not given)
8. Serialise the result as CML (`.cif.cml.xml`)

The process is not foolproof as CIFs do not always include the molecular charges correctly specified and any disorder may be difficult to interpret. The serialised CML produced should contain all the information specified in the CIF, but for display purposes a second file was created (`.cif.cml1.xml`) which contained only the molecules. The results of this process with the new file structure are shown in figure 7.5.

CIFs may contain multiple molecules in the unit cell (whether they are repeated instances of the same CT or different CTs). The workflow is designed to deal with each molecule individually, thus subdirectories were created for each of the molecules in the CIF and the relevant file created (`.inp.cml.xml`). These files contained the CT and 3D coordinates of the relevant molecule together with other data extracted from the CIF used to determine if they were suitable for calculation. The files can be easily recovered by searching for all files of filetype `.inp.cml.xml` under the `actae` directory. The final data structure adopted is shown in figure 7.6.

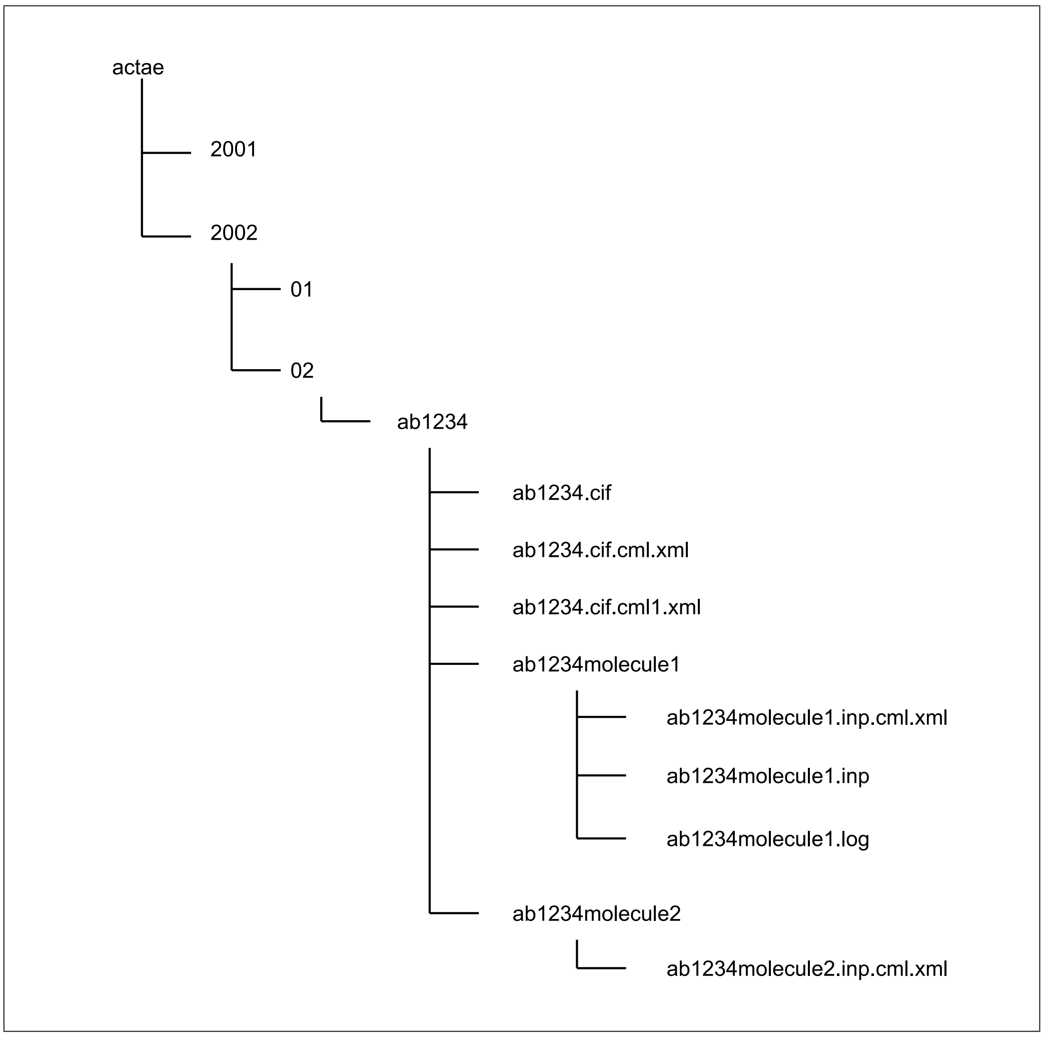


Figure 7.6: The hierarchical data structure adopted for the downloaded CIFs; the `moleculeN.inp.cml.xml` contains all the data relating to the Nth molecule found in the CIFs in CML. In this case `molecule1` was suitable for calculation (so the `.inp` file was created) whilst `molecule2` was not.

A traditional database has only one instance of a particular piece of information with many links to it — this is a *normalised* approach, whereas using the file system as described above promotes a *denormalised* approach. Each time a piece of information is required a copy of it is present (for example the atom site occupancies are in the CIF, the `.cif.cml.xml` and the `.inp.cml.xml` files) thus no links are required which removes a level of complication. The only relational ideas in the file system representation employed are that everything in a particular subdirectory is a finer-grained representation of what is the parent directory.

7.4 Create Input

Once a structure has been determined to be suitable for calculation, an input file must be generated for submission to the compute node or program. This file should contain the unique ID of the molecule. An ant [197] script was used to combine the coordinates from the `.inp.cml.xml` file with the job controls (determined by the protocol) creating the input file (`.inp`). The input files are then copied to submission nodes for computation.

To speed up the input creation process, the files for submission were split into two sets and the input creation script run on both concurrently. These files were submitted for calculation and many failed very rapidly. A brief examination of the log files showed that the correct input had not been created, for example certain keywords were missing. Further analysis revealed that some of the input files contained the coordinates for a different molecule. The entire set of jobs were cancelled.

The input file creation process was repeated and the files examined before submission to the compute nodes. The same type of mistakes were found in these files; although not always in the same files that contained errors the first time. The cause of the errors was found to be that the script used a temporary file to hold the data and the process was not designed to be multi-threaded. When run as a serial process the creation of input files succeeded.

It is desirable that a standard format to hold input data is implemented (*e.g.* CMLComp) with all the concepts linked to dictionaries [139]; these could either be read directly by the program or automatically transformed into program-specific input files as described in chapter 4.

7.5 Run

Once the input has been created, running the computation requires moving the requisite jobs to a suitable computation node and the job starting. This should be a trivial process but actually proved to be much more complex than expected. The calculations were to be performed on three separate clusters; Kellogg, Corona and Vendian.

7.5.1 The Clusters

Cluster is a widely-used term meaning independent computers combined into a unified system through software and networking. At the most fundamental level, when two or more computers are used together to solve a problem, it is considered a cluster. Kellogg is the home-built Beowulf cluster in the UCC, so-called because it runs only serial jobs. The cluster consists of seven single-CPU nodes, each with a 2.53GHz Pentium 4 processor and 1Gb memory. It is intended primarily for running long serial (single CPU) jobs; there is no parallel environment set up.

Corona is the name for the IBM x-series cluster at the UCC, intended primarily for running parallel jobs. The submission node of Corona is an IBM x-series 345, a 2U, dual-Pentium-Xeon machine with 4Gb of memory. The sixteen compute nodes are IBM x-series 335s, 2.4GHz dual-Pentium-Xeon machines with 4Gb memory. Although designed for parallel jobs, serial jobs may be run although they have very low priority.

The Condor pool, for which Vendian is the central manager, consists of this central manager, and 16 iPaqs. All the machines are setup as dedicated Condor hosts and are not used for interactive work. The central manager is a Dell OptiPlex GX150 (1GHz Pentium III, 512Mb memory) and the iPaqs have 500MHz Pentium III processors and 512Mb memory.

The jobs were apportioned between the clusters in the ratio $\frac{1}{5}$:$\frac{2}{5}$:$\frac{2}{5}$ respectively. The numbers of nodes stated above represent the maximum available on each cluster — the scheduler limits a user's allocation to allow other users computer time if required. Based on the data presented in table 5.3 the geometry optimisation of 1000 molecules containing 15 non-hydrogen atoms in would require a total run time of 1.1×10^8 seconds (3.5 years) to complete on average — which would take approximately 5 weeks of real time on the 39 available nodes. The actual real time required was over a year. Unfortunately, the Corona cluster suffered an unrecoverable crash not long after this work started resulting in the loss of 16 of the available nodes. The data, which was backed up, was recovered following which the uncalculated jobs were again reapportioned between the remaining available nodes and resubmitted.

The Vendian Condor pool was a recent addition to the UCC computing facilities following the success of the pool set up on the teaching machines (which had to be uninstalled). It is effectively a dedicated computing cluster which uses Condor as the scheduler and networking mechanism. The infrastructure was still being put in place to provide a stable environment whilst this work was being carried out which resulted in a very high percentage (>99%) of the calculations submitted failing. This was found to be caused by corruption of the GAMESS executable binary file whilst it was being sent to the compute node although this remained undiagnosed for some months.

Alternatives to the Vendian cluster were available in the form of the University-wide Condor pool set up as part of the CamGRID project [198]. Unfortunately this system, which uses Condor in the originally intended cycle scavenging way, does not provide guaranteed uninterrupted runtime and is only available between the hours of 10pm to 6am. This is because the University Computing Service will only allow the Linux based version of Condor to be used (because of security issues in the Windows based version) and the computers which make up CamGRID all run Windows but have a dual boot system. The computers are automatically remotely rebooted into the Linux operating system at 10pm (if they are not currently claimed by a physical user), when the Condor service becomes available and then rebooted into the Windows operating system at 6am. Of course, at any time between 10pm and 6am a user can reboot a machine into Windows if desired.

The lack on guaranteed uninterrupted runtime was not an insuperable problem because GAMESS has a checkpointing facility which allows the current state to be saved and the calculation restarted from that point. However, the remote rebooting of the machines does not allow the system to finish what it is currently doing which therefore does not allow the Condor system to retrieve the checkpoint data. Overall it was felt that camGRID did not provide a suitable platform for these calculations, therefore any uncalculated jobs (effectively all of them) were submitted to the only remaining resource, the Kellogg cluster.

7.5.2 Schedulers

When a job is submitted for calculation on a cluster, it is not being run directly, but entering a scheduler which actually performs the process of moving the job (together with any other specified files) to a particular node and starting the calculation. The scheduler may also implement a queuing system so that

> all the users of a particular cluster are treated equally badly. [199]

The scheduler provides both advantages and disadvantages; all users are treated fairly and the submission of a large number of calculations *en masse* is possible but it is a further level of complexity which can crash. In cases where the scheduler crashed, all the completed jobs were removed from the cluster for parsing and analysis and those which were either unsubmitted or incomplete were reentered into the queue.

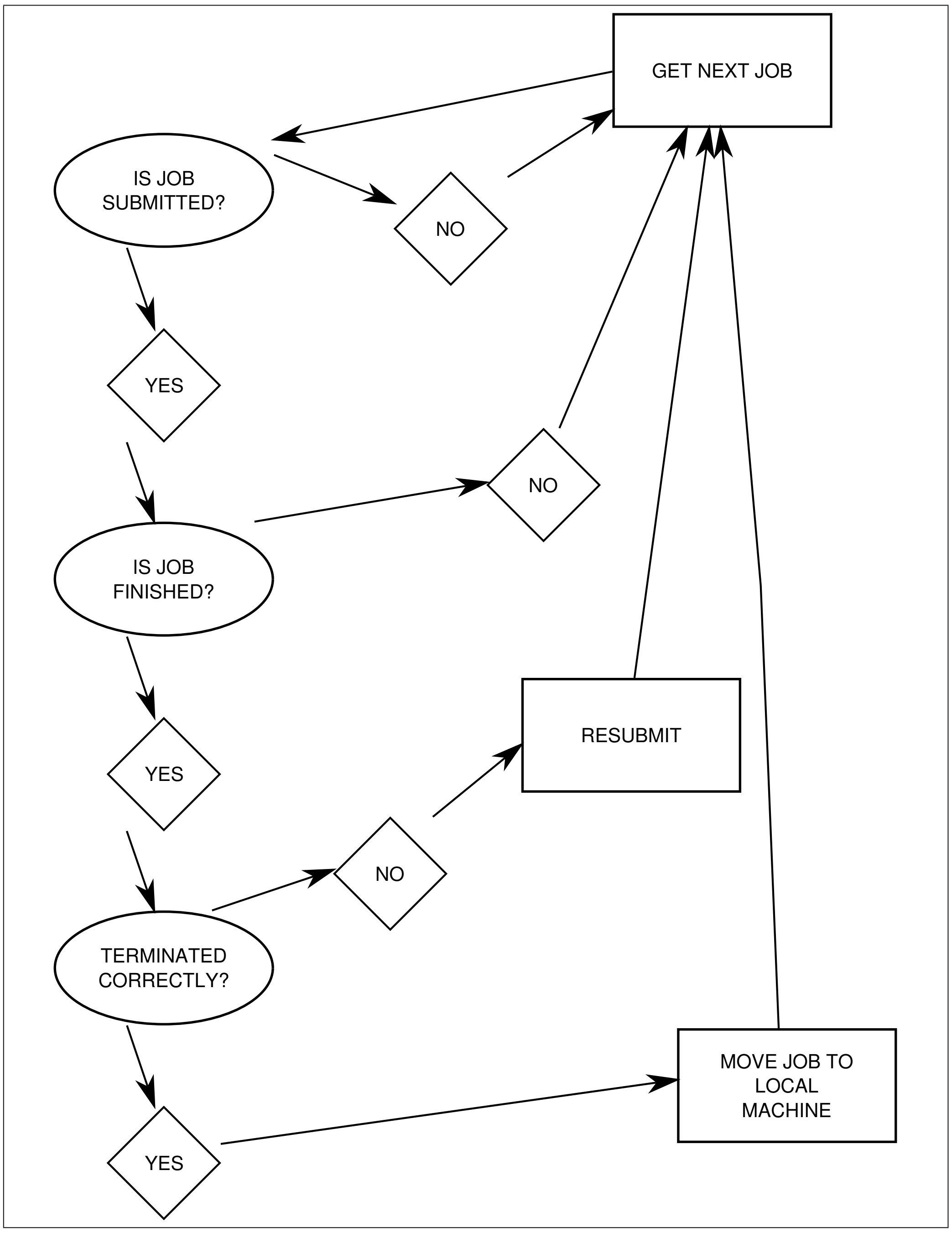


Figure 7.7: The decision diagram for retrieving jobs. Ovals represent decisions, each of which requires a separate method, rectangles represent simple instructions or other workflows. It should be noted that this diagram is simplified.

7.6 Retrieve Results

This process is actually a separate workflow in its own right. Figure 7.7 shows the steps required for this sub-workflow. The *rules* governing the three decisions are shown below.

Is Job Submitted? Although the submission script adds all the jobs to the queue, the actual calculation may not have started. The submission for a job consists of two files:

> **Input File (`.inp`)** This consists of the starting coordinates of the atoms and control information for GAMESS.
>
> **Submission File (`.sub`)** This consists of a set of instructions for the node running the job such as where the input file is located, where the GAMESS program resides and what to do with the various outputs.
>
> If the job is not in the queue but these two files (and only these two files) are present, then the job has not been submitted and should be added to the queue. If the job is in the queue then it simply has not begun to execute and no further action should be taken.

Is Job Finished? Once a job has begun executing, four further files are created (the `.inp` and `.sub` files must already be present for execution to occur):

> **Log file (`.log`)** The information that would usually go to 'System.out', designed to be slightly human-understandable.
>
> **Data file (`.dat`)** Essentially a more compressed version of the data that is sent to the log file. Less human-understandable and also more difficult to machine-parse completely unambiguously. The start and end points are less well defined. This file only contains the data, rather than fail messages.
>
> **Queue manager log file (`.pbs.log`)** An empty file as all the information is sent to the error file.

Queue manager error file (`.pbs.err`) A complete list of where GAMESS stores particular files *etc.*

Any job that has only these six files present is treated as finished and is suitable for further processing. However, GAMESS also creates a series of F files. These all have file types of the form:

```
.F\d\d?
```

The presence of the these F files indicates that either the job is still running, or that the node has crashed during the execution of the job. Unfortunately such information is not present in the `.pbs.err` file. It is thus necessary to interrogate the list of currently executing jobs to determine if this job is present. If it is then no further action should be taken, if it is not then the job has crashed during execution and this should either be immediately resubmitted automatically or moved to another location ready for manual resubmission. Although there is a desire to automate the entire process, it may not be sensible to have automatic resubmission of jobs of this sort because an analysis of the failed job might lead to useful knowledge.

If finished then move else wait If the job has finished then remove the files to a permanent store otherwise look at the next job, if all the jobs have been looked at then wait for a specified time before restarting the process.

Determine if job has terminated correctly JUMBOMarker was used to determine that the log file produced was complete as described in section 4.3. If the log file is complete then the file can be moved to the results repository, otherwise it should be resubmitted.

7.7 Results Repository

Web-based repositories do exist, such as DSpace, which captures, stores, indexes, preserves, and distributes digital research material [200]. However, whilst analysing the data produced, a local repository was preferred, because until the analysis was completed the files containing the data were being frequently accessed and modified. A local repository reduces access times, bandwidth requirements and removes a level of complication from the analysis.

7.8 Designing a Robust Analysis Method

Once the calculations have been performed and the data is in the local repository, the majority of the analysis can occur. Chapter 5 details various examples in which the calculation was changed owing to peculiarities discovered in the data created, and such changes were also necessary in this study. This involved finding molecules which were inappropriate for calculation and *tightening* the protocol so that such instances would be filtered off before submission (see section 7.11).

Increasing the severity of the filters imposed on the data for calculation is useful for increasing the efficiency and validity of future calculations but not for those molecules already calculated; these must be retrofitted with the improved protocol. Two possible ways of retrofitting the data are

- altering the file structure
- altering the file

7.8.1 Altering the File Structure

Altering the file structure could range from changing the name of a file (so that it would no longer be included in an iterative call for all files of a particular type), to the complete deletion of a file and all data derived from it. The second of these examples would decrease the amount of storage required for the data and the subsequent corpus would more closely resemble the data structure generated in future using the protocol.

It was felt that the needless and wanton destruction of data in this manner should be avoided and therefore all results (however ridiculous) should be kept in the dataset and eventually archived along with the rest. However, included in the aims of this study are a desire to check the validity of the data available in the public domain, to allow re-use of the data and to allow others to recreate experiments, thus it is necessary to provide an indication of the quality of the data during this process together with the necessary tools to analyse it.

7.8.2 Altering the File

Altering the file is a logical progression from the denormalisation of data; all the data used to determine whether or not a molecule in a particular file is suitable for calculation should be present in that file. The files already contain elements or attributes from the original CIF indicating, for example, the presence of disorder; further elements were included during analysis to indicate whether or not a molecule passed a particular filter, or the data required to ascertain if it would pass.

The elements added to a file need not be those pertaining to the requirement of the protocol. For instance if a molecule was found to be protean this can also be indicated. Elements added purely to determine if the molecule has passed the requirements for the protocol, or is suitable for further analysis are in the form;

```
<scalar id='flag' dictRef='jat:cyclic_flag'>ACYCLIC</scalar>
```

or

```
<scalar id='flag' dictRef='jat:proteus_flag' />
```

and are referred to generally as flags.

7.9 Creating a Protocol

When performing HT calculations it is important to reduce, as far as possible, the error rate of those calculations and also to maximise the amount of useful work done in the time available. Previous experience of designing protocols, and analysis of the calculations performed suggested that it is a better use of the available resources to construct a fairly simple protocol and refine it after calculations have been performed rather than attempting to design the perfect protocol before starting any calculations. It is more efficient to perform more radical but crude filtration than to create very specific filters; although this might prevent a few *interesting* structures from being calculated, overall, more structures will be determined.

A protocol may be divided into two separate sets of parameters:

Job controls such as the time allowed, the memory available, the size of the basis set and the level of theory to use.

Molecule selection is it reasonable to attempt to calculate the geometry of this molecule given the specified job controls?

The job controls used were those which had been determined as a result of the analysis of the MOPAC molecules, *i.e.* the 6-31G* basis set, B3LYP exchange function, internal coordinates and a maximum runtime of one week (an example of a full input file showing all the job controls is shown in figure 5.7).

7.10 Molecule Selection Parameters

Molecules that are attractive for submission for both high- and low-throughput computation are ones that will provide interesting results and that run to completion rapidly. Finding such molecules is both difficult and time consuming. Whilst low-throughput computation might focus more on a few *interesting* molecules and allow such calculations to run for extremely long times, HT computation focuses on getting as many results as possible in the given time.

The initial parameters selected to determine if a molecule was suitable for calculation were

- no disorder
- element type
- number of non-hydrogen atoms

Disorder in CIFs can present a problem, especially when the disordered groups are not correctly identified, or the occupancy of the disordered sites are equal (because then it is not trivial to determine which of the sites go together to form each group). To prevent such problems, any structure that contained disorder was determined to be ineligible for calculation — it should be noted that this check was performed on a per-molecule basis rather than a per-CIF basis, hence some molecules in a CIF might be eligible for submission whilst others are not.

This study focussed primarily on organic structures with well-described bonding schemes containing only those elements which can be represented using the basis set and level of theory stated above. Therefore only molecule containing the following element types were considered for submission: H, B, C, N, O, F, Si, P, S, Cl and Br. Table 5.3 shows the predicted run time required for a calculation involving a certain number of non-hydrogen atoms. This suggests that ca. 95% of the calculation on molecules containing 20 non-hydrogen atoms are likely to complete within a week. Molecules containing fewer than 20 non-hydrogen atoms are likely to run to completion in less time.

To prevent calculating the structures of too many solvent-type molecules (for example water and dichloromethane) which, because of their abundance in the CIFs, would dominate the available computing time and provide little of interest, a minimum number of non-hydrogen atoms was also imposed. The minimum value chosen was four, this removes many small solvents and counter ions (such as ammonium) without preventing some of the more interesting small molecules from being calculated (especially α substituted carbonyl groups with one to three fluorine and chlorine atoms present).

The imposition of the initial parameters on the dataset yielded ca. 2400 molecules. It was decided that this should be split into two sets of input data: those molecules with four to 15 non-hydrogen atoms, and those with 16 to 20. The sets contained approximately 1200 and 1000 molecules respectively. The submission of the latter set (with more non-hydrogen atoms and hence longer run times) was dependent on the results derived from analysing the first set. Unfortunately, although the results of the initial dataset were positive, the loss of available compute resources and time constraints prevented the second set from being calculated. Therefore the following calculations consider molecules containing four to 15 non-hydrogen atoms, with no disordered atoms present and only containing the elements specified above.

It was fully expected that further refinement of the selection parameters would be required as the analysis progressed. Methods for implementing these are described in section 7.8. A filter to remove molecules with unpaired electrons was not implemented because it was thought that the limit imposed on the number and type of atoms would make it impossible for such molecules to be present. Only very stable radicals are sufficiently stable to form crystals which are typically found in large structures or structures containing heavy metals [201].

The total number of CIFs in the corpus was 6738, from which 6676 were successfully parsed to CML. The 63 which were not parsed lacked particular required data items or contained the data items in a form which was not understandable by the parser (they did not conform to the CIF dictionary within acceptable tolerance). The parsed CIFs yielded a total of 6455 molecules. Using the three initial parameters 1181 molecules were found that were suitable for calculation.

7.11 Refining the Protocol

The comparison of MOPAC and GAMESS bond lengths suggested that performing geometry optimisations at 6-31G* with GAMESS provides consis-

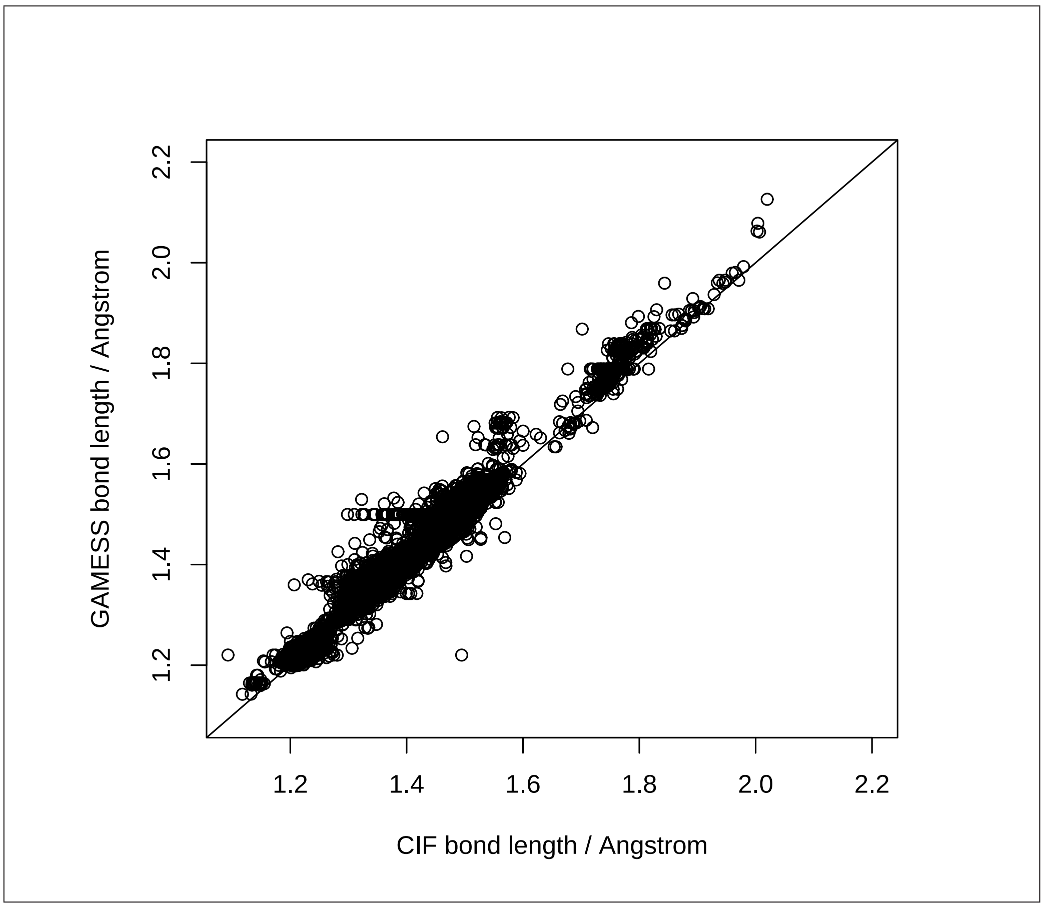


Figure 7.8: One clear outlier is visible, as is the appearance of bands, for example at a calculated bond length of 1.5Å. The $x = y$ line is shown.

tent, and high-quality data. Crystallographic data is considered to be of a consistent quality (as opposed to the NCI database), therefore any outliers found between the bond lengths reported in the CIF and those calculated by GAMESS are indicative of possible problems in either the experimental or calculated values — or that like is not being compared with like. All bonds to hydrogen atoms are excluded from the analysis because their positions are poorly determined by crystallographic methods in general. Proteus molecules were also removed from the dataset before the geometries were compared. The analysis of these molecules is presented in section 8.2. As before, both x-y and QQ plots were used to determine outliers.

Figure 7.8 shows all the 9512 bond lengths reported in the CIF against those calculated by GAMESS (excluding bonds to hydrogen); the bonds are from 973 molecules. There are two features of note; a clear outlier and an apparent banding. The molecule giving rise to the outlier was examined and found to be *N*,*N*-dimethylformamide. The bonds giving rise to the banding at a GAMESS bond length of 1.5Å were all found to be perchlorate ions.

The determination of the positions of atoms in solvent/ion/guest molecules (such as the perchlorate anion) is not given the same priority as those from the major structure and therefore these molecules should not form part of the analysis. A program was written to check whether or not a molecule was a solvent/ion/guest and to add a flag indicating the results of this test. Appendix F shows the list of molecules, or molecular fragments considered to fall into this category. Figure 7.9 shows the QQ plot of all the 9512 bond length changes — the outliers are not as clearly apparent but there appear to be overlapping sets of data, with the behaviour changing at $\Delta R \approx \pm 0.05$Å. The cause of this is observed later.

After the solvent/ion/guest molecules were removed from the dataset, 8728 bonds (from 785 molecules) remain; the QQ plot of these is shown in figure 7.10. The major outliers are caused by the loss of hydrogen bonding (on moving from the crystalline state to a single molecule in vacuum). Effectively

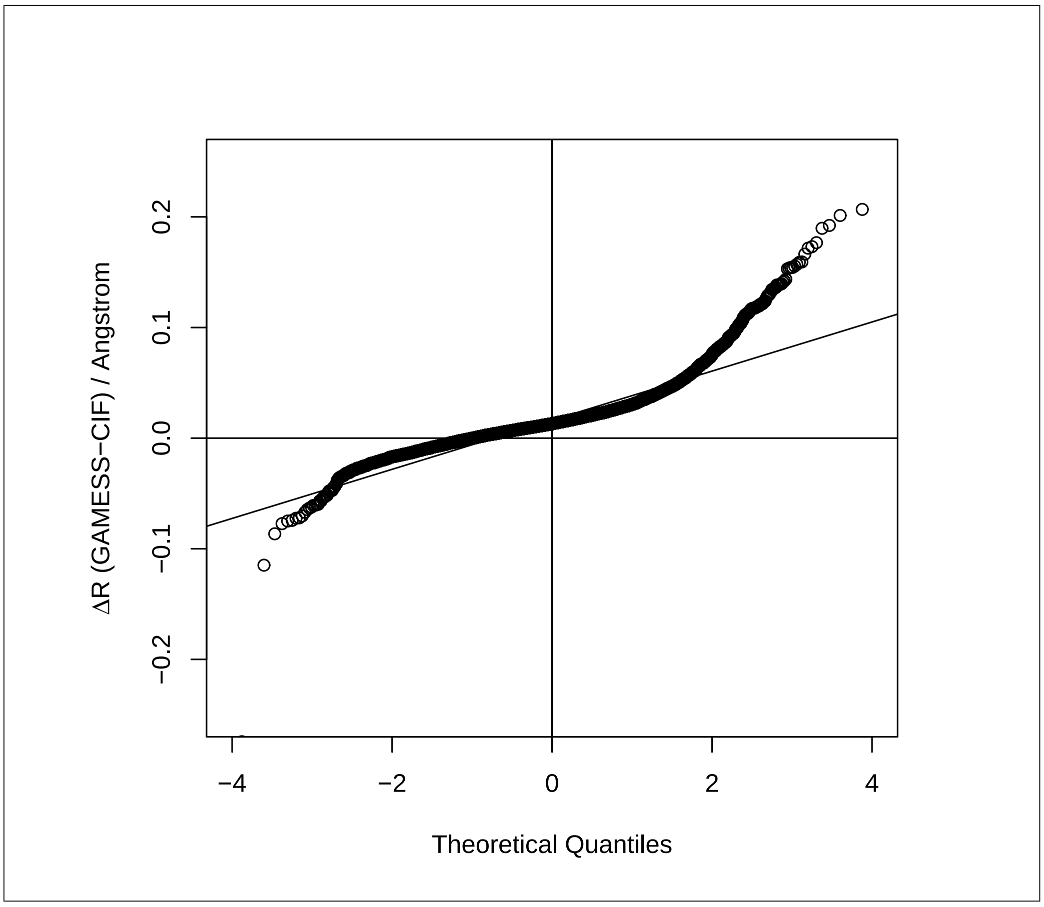


Figure 7.9: All bonds in the corpus excluding those to hydrogen — the data appear to be normally distributed within $\Delta R \approx \pm 0.05$Å. It is possible that there are several overlapping distributions. Overall $\overline{\Delta R}$=0.016Å and s=0.022Å.

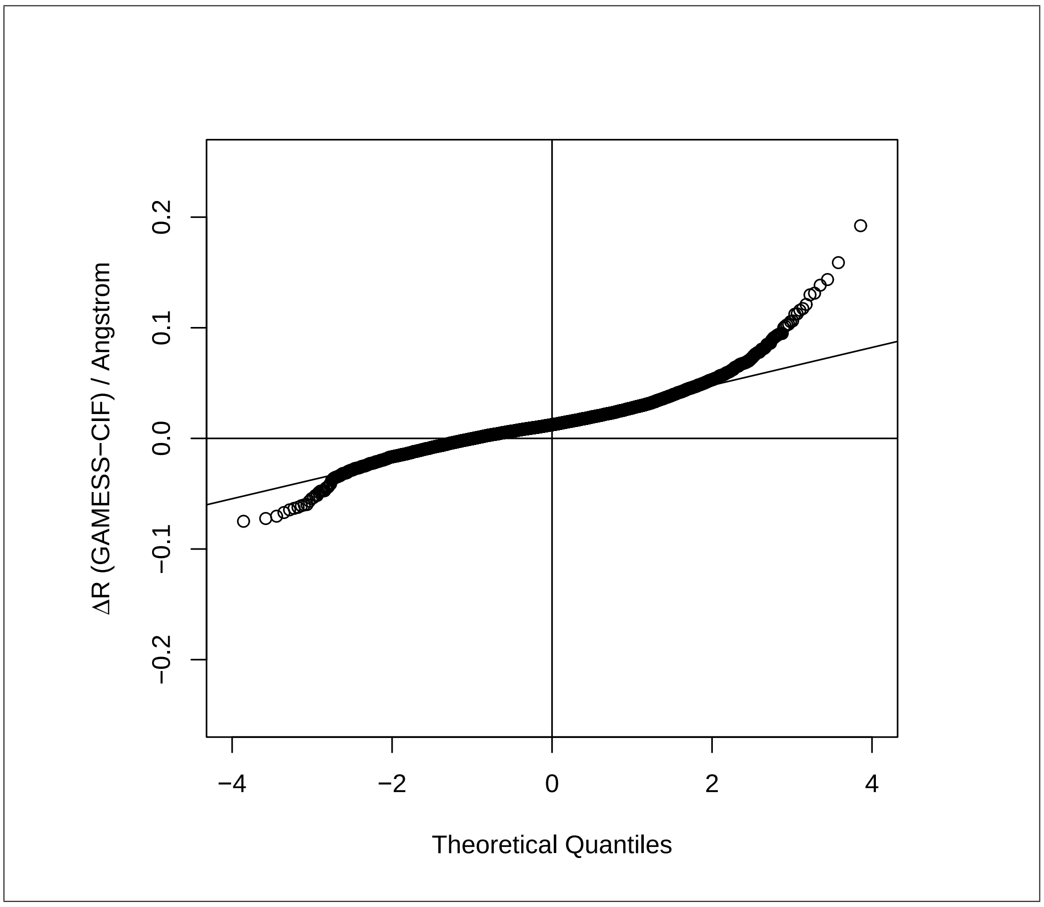


Figure 7.10: The QQ plot of all bonds excluding those from solvent/ion/guest molecules — the data is mostly normally distributed with $\overline{\Delta R}$=0.014Å and s=0.017Å.

this means that the comparison of bond lengths is not comparing like with like, resulting in large outliers. To remove such outliers from the dataset, all bonds involving oxygen atoms that were not bonded to two non-hydrogen atoms, are marked with a flag to indicate that they might be affected by hydrogen bonding and removed from further analysis. Similarly all nitrogen atoms that are bonded to one or more hydrogen atoms and are sp^3-hybridised may be hydrogen bond donors and are therefore also flagged. It should be noted that this is done on a per-bond basis, thus other bonds in the molecule are still included and the molecule would still be considered suitable for calculation.

After the removal of possible hydrogen bonding effects, 7177 bonds (from 782 molecules) remain; the QQ plot of these (figure 7.11) shows that there are still outliers. On examination many of the outliers were caused by molecule with large R-factors; following advice from Alison Edwards, all CIFs with R-factor > 0.05 were removed from the dataset, and the protocol altered to filter off such structures in future because these are less likely to be high quality structures [202]. Figure 7.12 shows the QQ plot of the 5034 bonds from 571 molecules following the imposition of this filter.

It is possible to place constraints on particular atom sites during the determination of the crystal structure which are labelled in the CIF using the `_atom_site_refinement` data item. It was felt that structures with constrained atoms (except hydrogen) were not sufficiently experimentally determined.

The initial filtering of constrained atoms was performed on a per-bond basis — only those bonds containing constrained atoms were removed from the analysis, because the remaining bonds in the molecule would still reflect experimentally determined values — but this proved to be ineffectual. Therefore, all molecules containing constrained atoms were removed from the dataset. Figure 7.13 shows the QQ plot resulting from removing only constrained bonds rather than the entire molecule (4264 bonds from 547

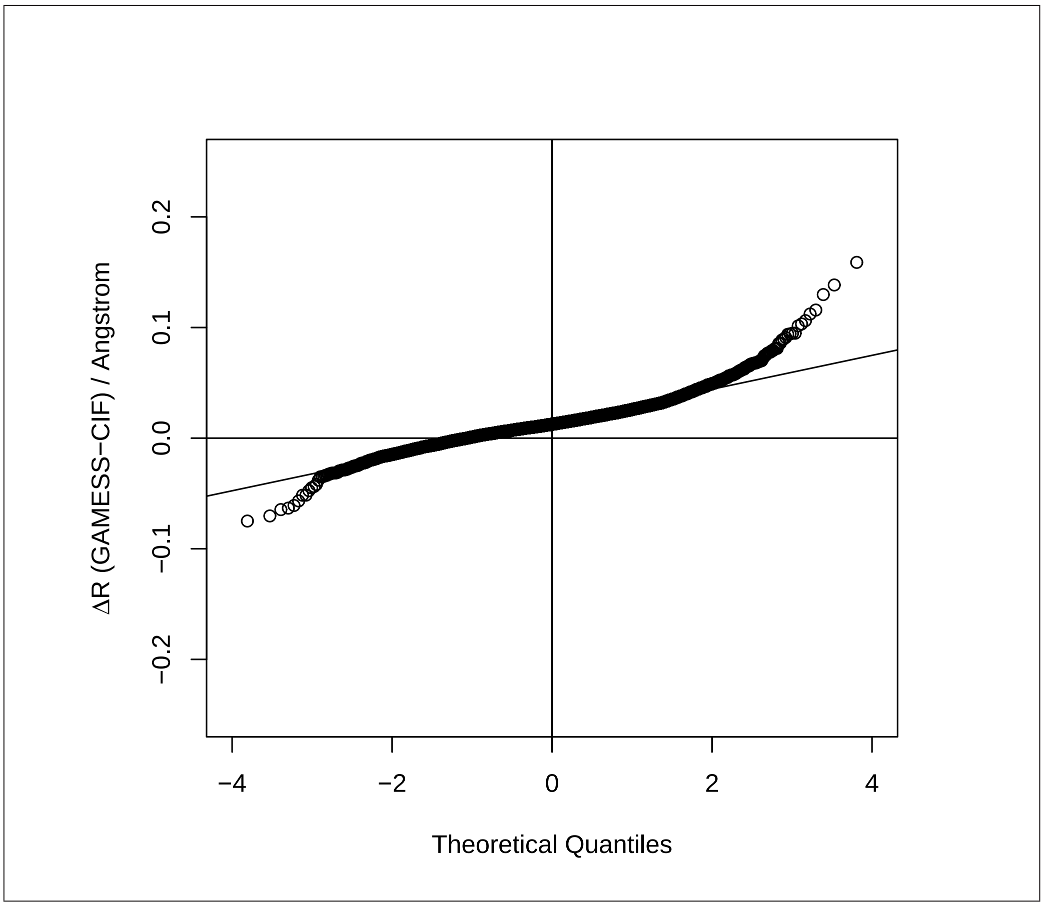


Figure 7.11: The QQ plot of 8728 bonds excluding those from solvent/ion/guest molecules and those which are possibly effected by hydrogen bonding — the data is mostly normally distributed with $\overline{\Delta R}$=0.014Å and s=0.015Å.

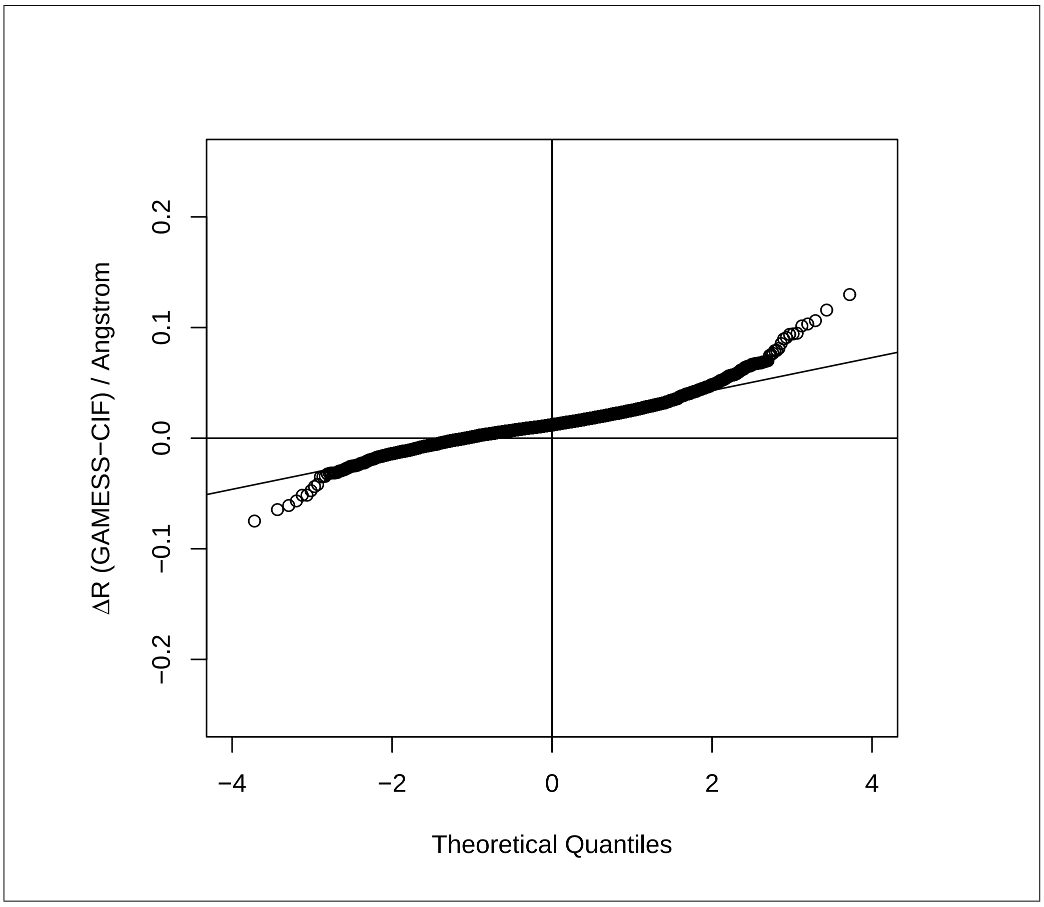


Figure 7.12: R-factor $\leqslant 0.05$ — the data is mostly normally distributed with $\overline{\Delta R}$=0.013Å and s=0.015Å.

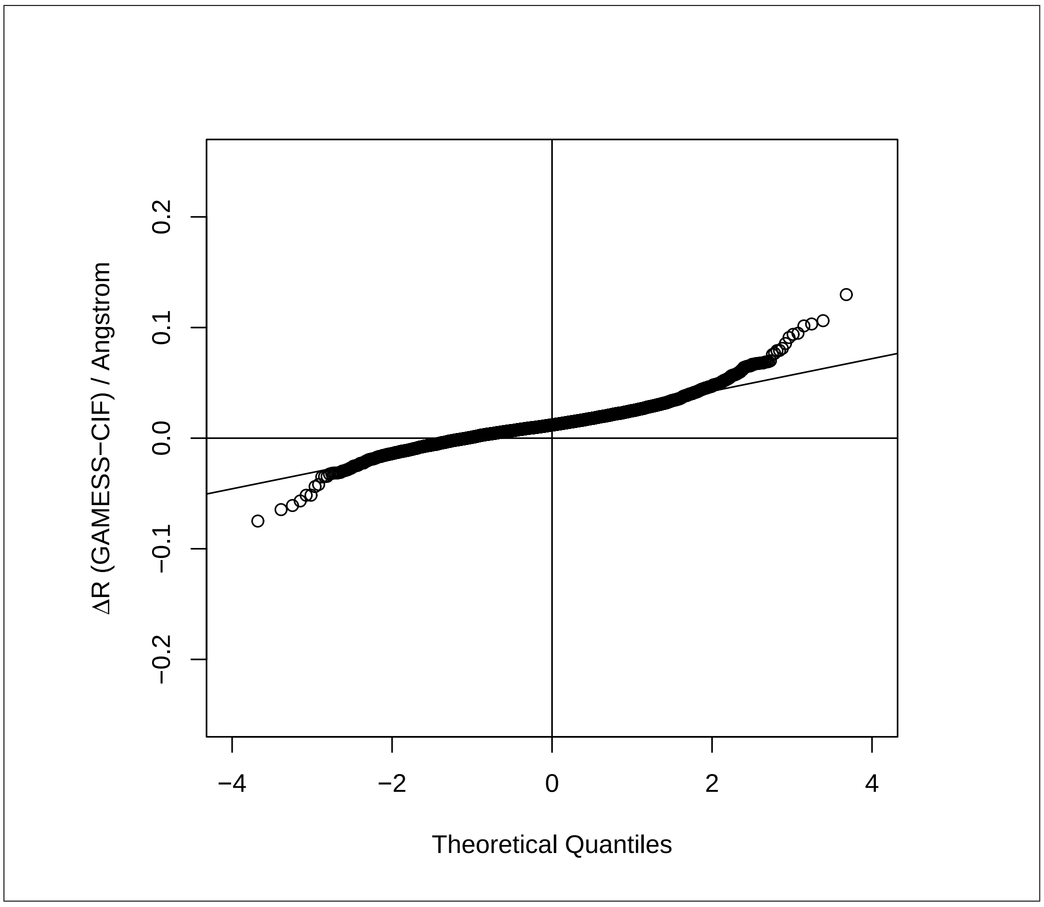


Figure 7.13: No constrained bonds — the data is mostly normally distributed with $\overline{\Delta R}$=0.013Å and s=0.015Å.

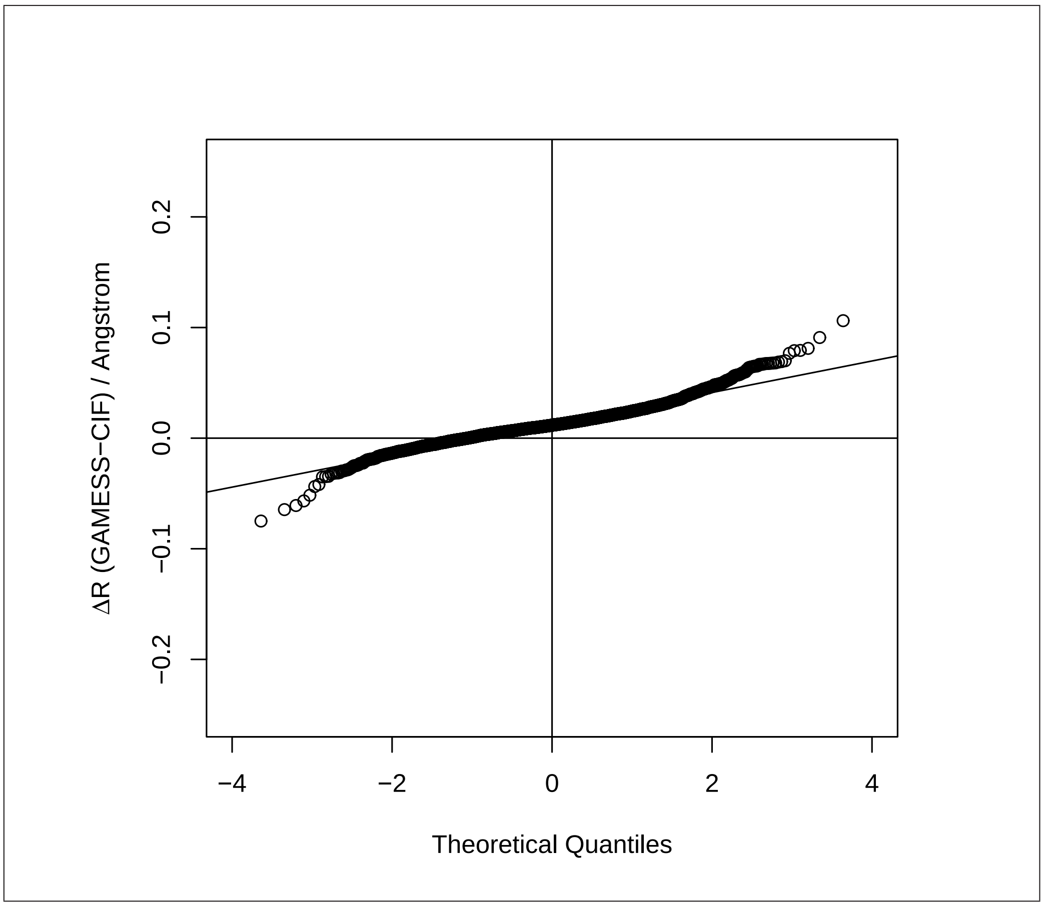


Figure 7.14: No molecules containing constrained non-hydrogen atoms — the data is mostly normally distributed with $\overline{\Delta R}$=0.013Å and s=0.014Å.

molecules).

Figure 7.14 shows the QQ plot of the 3658 bonds (547 molecules) which remain following the imposition of the filters described above. The outliers $\Delta R < -0.05$Å have two causes; loss of aromaticity and incorrectly specified charges in the CIF. 1,3,5,7-cyclooctatetrene *in vacuo* has been calculated to have alternating bond lengths of 1.47Å and 1.34Å reflecting the non-aromatic nature of the molecule. The crystal structure contains the molecule complexed with thulium allowing metal-ligand electron donation. The reported C–C bond lengths in the CIF are between 1.40Å and 1.42Å reflecting the

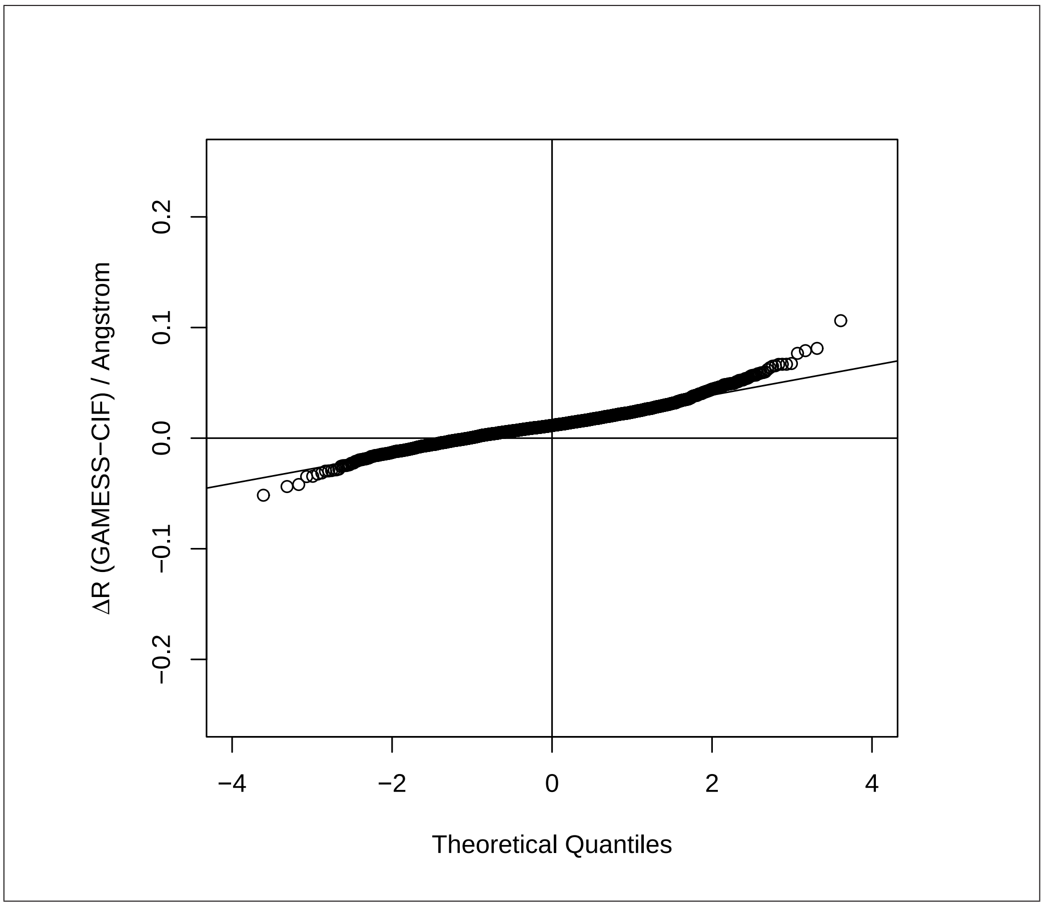


Figure 7.15: Only molecules from CIFs containing the specified elements — the data is mostly normally distributed with $\overline{\Delta R}$=0.012Å and s=0.013Å.

aromatic nature of the bonding in this state. There is no reason to believe that the bond lengths obtained from the CIF and the GAMESS calculation are incorrect but they describe the molecules in different states. The outlier should therefore not be interpreted as being caused by bad data.

The process of determining which atoms are part of a molecule is not currently sufficiently chemically aware. This frequently results in coordinated molecules and organometallic molecules becoming separated (for example ferrocene will become two cyclopentadienyl molecules and an iron atom). The presence of heavy atoms (in the crystallographic sense) in the crystal also reduces the accuracy of the determination of the positions of the lighter nuclei. Therefore CIFs which contained other than the permitted nuclei (H, B, C, N, O, F, Si, P, S, Cl and Br) were removed from further analysis. Figure 7.15 shows the QQ plot of the 5253 bonds (348 molecules) that fulfil this requirement.

The CIF containing the molecule with incorrectly specified charge contained two charged molecules and the authors had mistakenly swapped the charges over. There is no simple algorithmic method for determining that this has occurred, so a manual removal flag was used. 5251 bonds (347 molecules) remain in the dataset; the QQ plot of these is shown in figure 7.16.

The temperature at which the crystal structure is determined affects the quality of the resultant structure, as mentioned in chapter 6. Traditionally structures were resolved at room temperature but the introduction of cryocooling has allowed low temperature studies to be performed. The temperatures reported in the entire corpus for this work ranged from 5K to 573K (the upper bound is highly dubious — see section 8.5). Following advice from Bernd Schweizer all crystal structures with a reported temperature greater than 200K were removed and a filter created to prevent the calculation of such structures in future [203]. Figure 7.17 shows the 1397 bonds (150 molecules) still in the dataset.

To more completely remove the crystal packing effects from the analysis, only bonds which are part of a ring are considered. This involved creating a flag for each bond to designate whether or not it was a cyclic bond — this is has the sole effect of minimising the crystal packing effects on the analysis of the data; molecules containing no cyclic bonds are still suitable for calculation.

990 bonds (128 molecules) remain after the implementation of all the filters (including that for cyclic bonds). Figure 7.18 shows the QQ plot of the bond

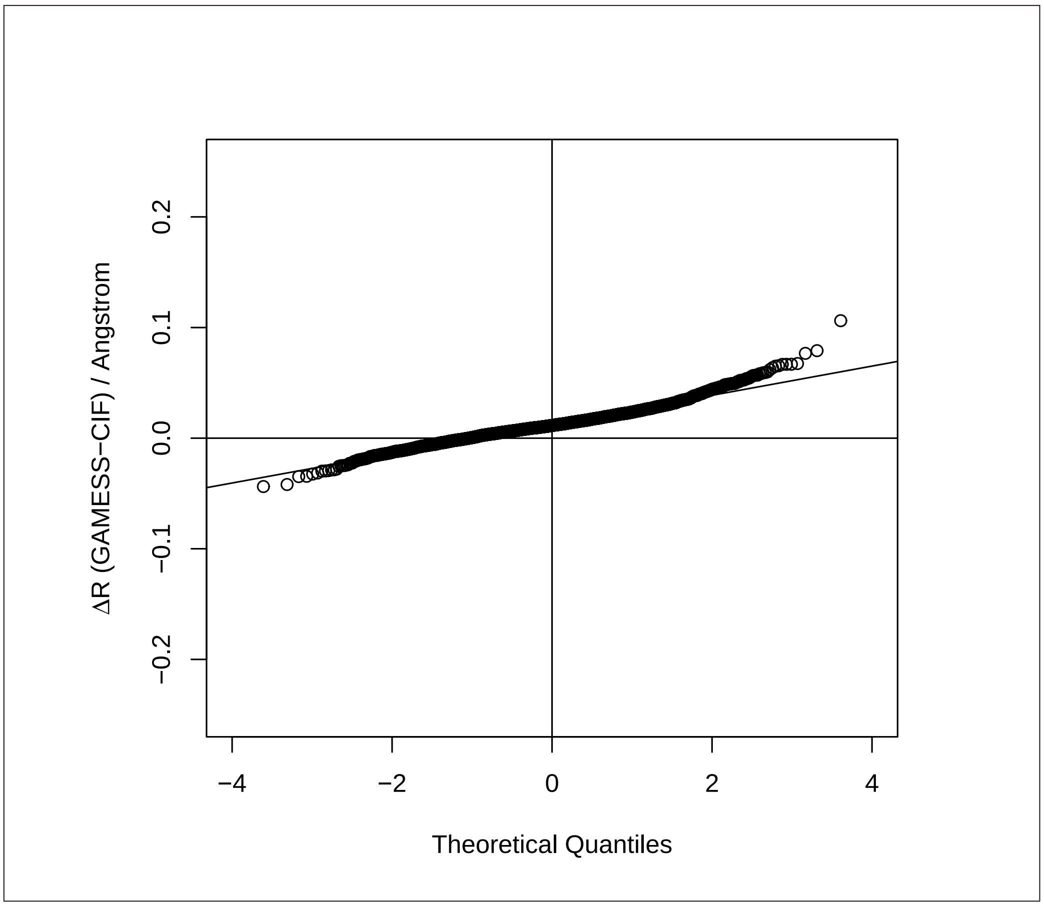


Figure 7.16: Manual removal — the data is mostly normally distributed with $\overline{\Delta R}$=0.012Å and s=0.013Å.

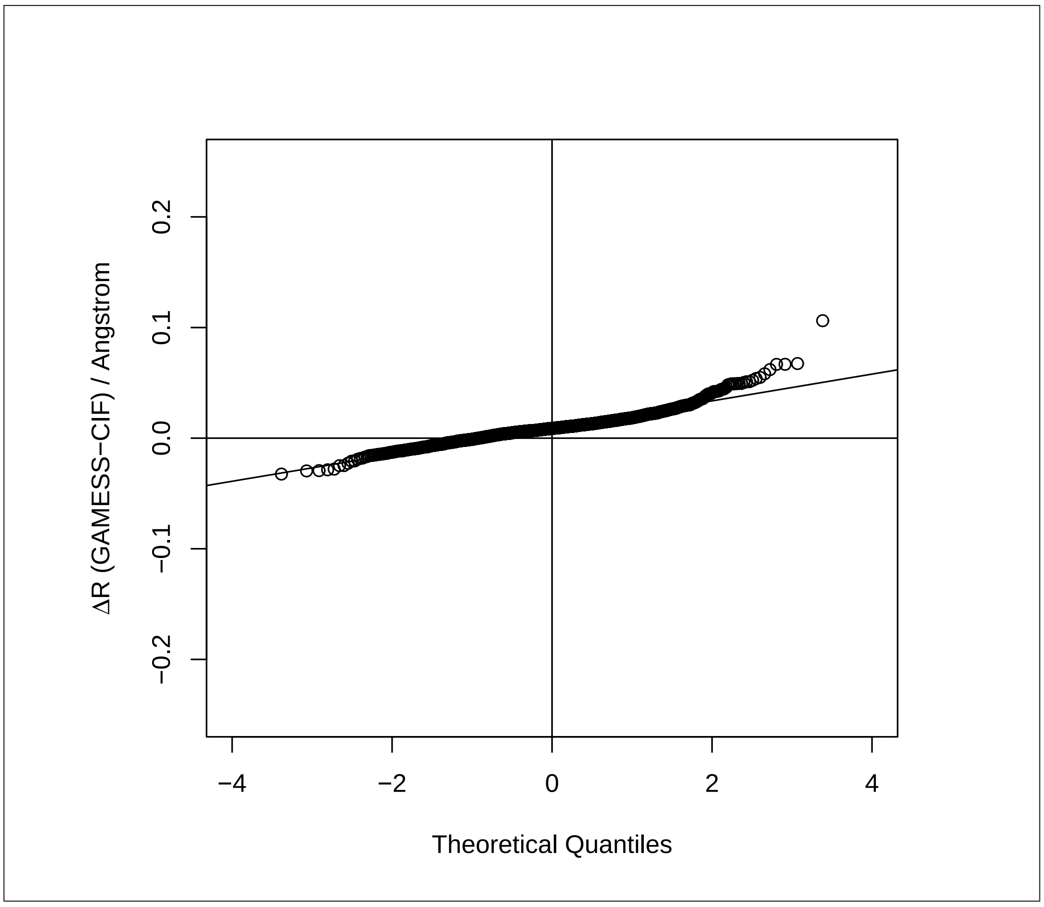


Figure 7.17: T $\leqslant$ 200 K — the data is mostly normally distributed with $\overline{\Delta R}$=0.009Å and s=0.012Å.

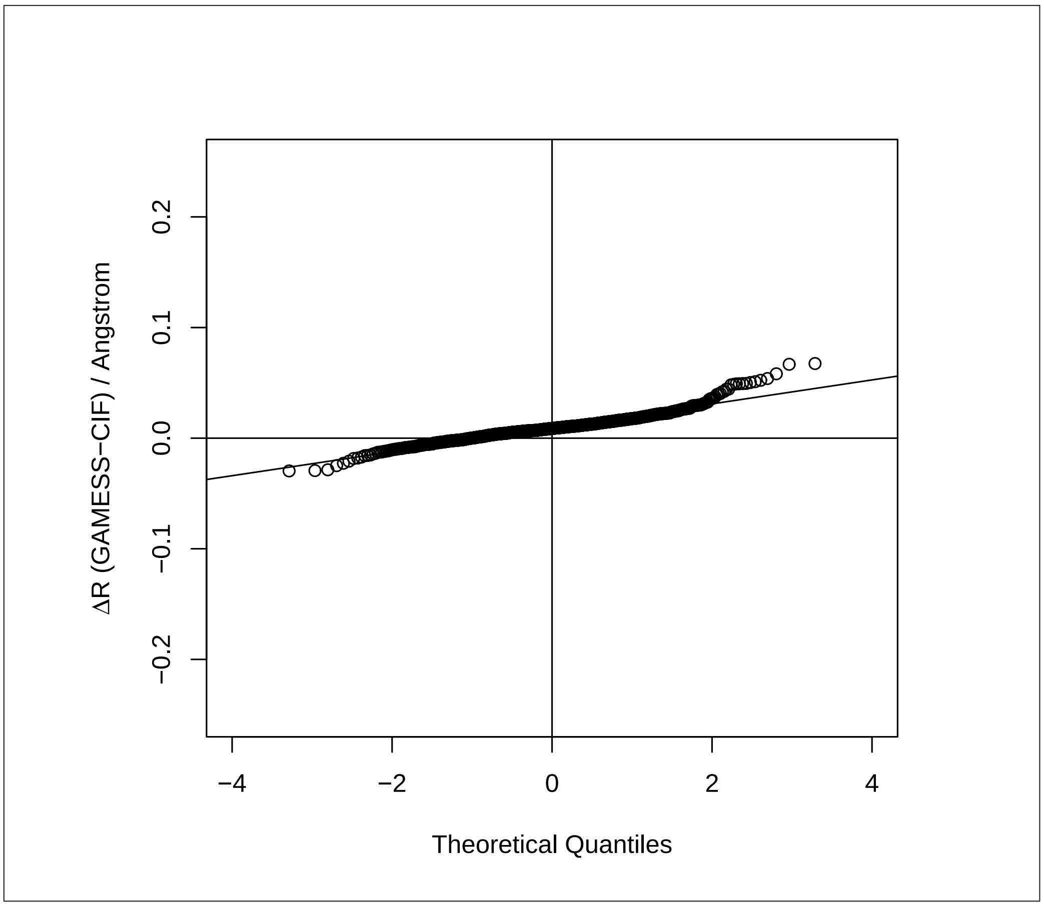


Figure 7.18: Cyclic bonds only — the data is mostly normally distributed with $\overline{\Delta R}$=0.009Å and s=0.011Å.

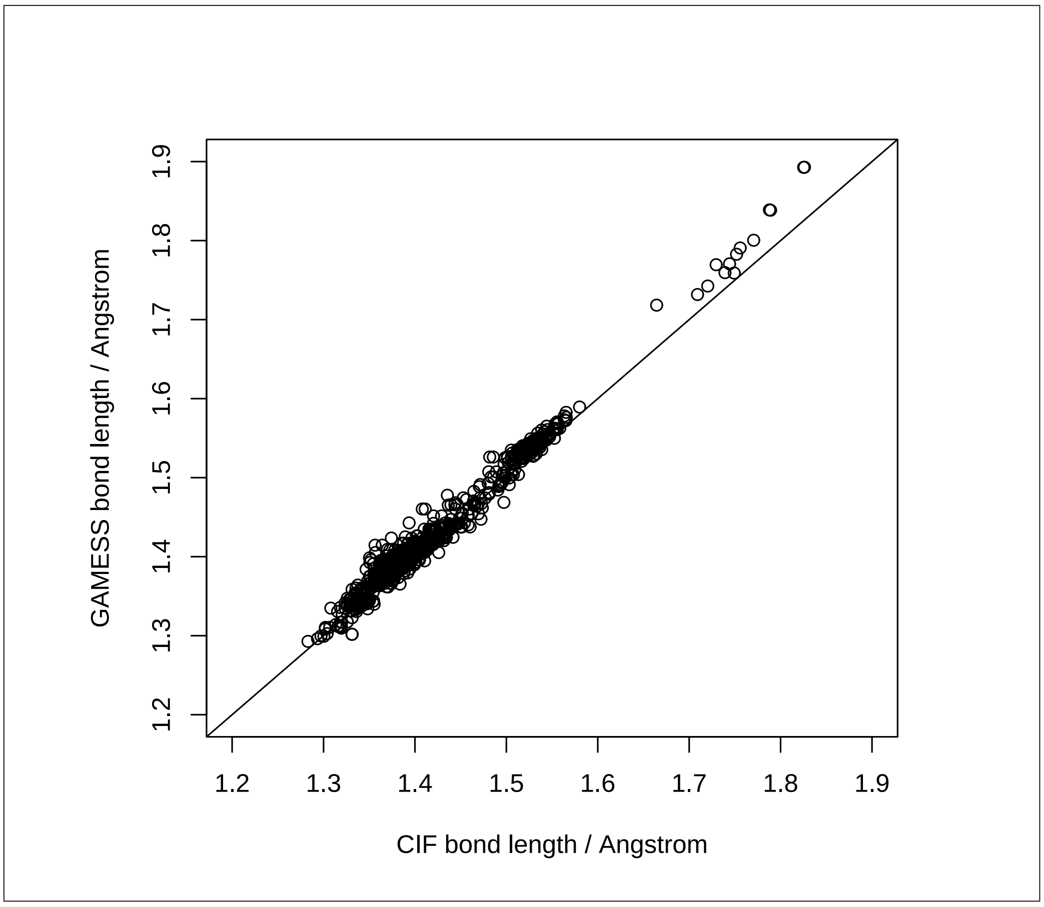


Figure 7.19: The calculated bond lengths appear to be consistently longer than those determined experimentally, with the effect becoming more pronounced for longer bonds. The $x = y$ line is shown.

length comparisons. There is a slight positive skew present — GAMESS bond lengths are found to be consistently longer than the corresponding CIF bond length — with this effect becoming more pronounced for longer bonds. The lengthening is more apparent in the x-y plot (figure 7.19) and is examined in chapter 8. Table G.1 shows the 112 unique connection tables of the 128 molecules which pass the final protocol.

7.12 Conclusions

As expected, the analysis of the data led to further filtering and refinement of the original protocol. The final protocol is given below:

- no disordered atoms
- molecules from CIFs consisting only of the following element types are permitted: H, B, C, N, O, F, Si, P, S, Cl, Br
- molecules containing more that 4 non-hydrogen atoms are suitable — but the more of these atoms are present the longer the run time is likely to be (see tables 5.3 and 8.10).
- solvent/ion/guest molecules are not suitable
- only molecules with a R-factor (`_refine_ls_r_factor_gt`) $\leqslant 0.05$ are suitable
- molecules containing constrained atoms (`_atom_site_constraints`) other than hydrogen atoms are not suitable
- structures determined at a temperature greater than 200K are not suitable
- manual removal of author error

A further two factors should be considered when comparing the reported bond lengths in the CIF and those calculated *in vacuo*. Firstly, hydrogen bonding in the crystalline form can affect the bond lengths of the atoms involved and adjacent bonds (see section 8.4.3); the most common hydrogen bond acceptors and donors found in the dataset were carbonyl and imine groups, alcohols and amine groups respectively. Algorithms to detect bonds to these groups have been created and implemented. Secondly, the effects of crystal packing forces can largely be negated by considering only bonds which are part of a cyclic system. The identification of cyclic bonds is possible using the current tools and has been implemented.

Developing a suitable architecture for HT computing is non-trivial. Workflows must be developed with flexibility and tolerance to failures in mind. The protocol should be expected to undergo considerable refinement before it can be considered suitable. This refinement is likely to be directed by the analysis of the results; to analyse the data produced without knowing the limitations of the protocol is meaningless.

Traditional workflows programs (such as Taverna) are not well suited to the kind of HT computing detailed above. It *might* be possible to incorporate an entire workflow in such a program but to do so increases the complexity of the code required. Much of this complexity results from the limited input and output format for each of the processes in such programs. The program writers are therefore forced to implement the necessary algorithms to perform the intended process as well as the transformation of the data to and from the specified workflow format. Traditional workflow programs are also not designed to implement processes on multiple platforms; this is often an absolute necessity in HT computing.

In general, it is the belief of this author that workflows should be constructed from simple, lightweight components each of which should perform one process. Each process should be written in such a way that it is as general as possible and clearly reports any errors encountered. Passing data between the processes is most easily accomplished by correct use of the file system; a denormalised approach (possibly with each process creating a new instance of a file) is desirable. The processes may be linked together using a script if desired. Even if each of the processes are encapsulated in a WS and made publicly available they are unlikely to be globally usable although local re-use is possible. Examples of processes that may be re-usable locally are input creation, submission and retrieval of results.

Chapter 8

Results

Some analysis of the results has already been presented in the previous chapter. This analysis was necessary to determine how the original protocol should be modified and only considered non-proteus molecules that had completed the geometry optimisation. To give a complete picture of the applicability of the workflow, the failed calculations and the proteus molecules must also be examined.

8.1 Failure Analysis

The 1181 jobs submitted for calculation under the original protocol yielded 180 instances where the calculation failed (see table 8.1). There were four sources of failure:

- insufficient time to finish the minimisation (21)
- bad delocalised coordinates generated (16)
- SCF did not converge (78)
- incorrect charge and/or multiplicity specified (65)

The molecules giving rise to each of these failures were examined to ascertain whether it would have been possible to create a filter to remove jobs of this type before submission.

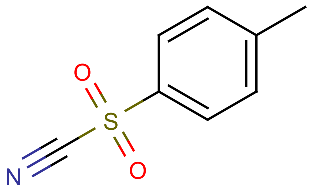
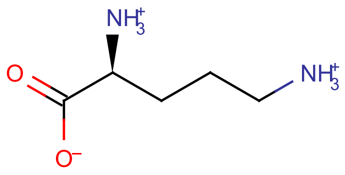
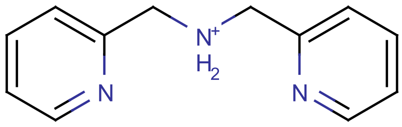
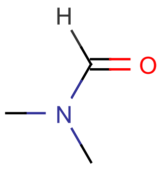
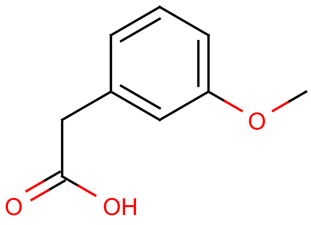
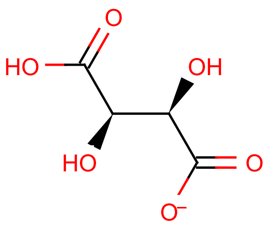
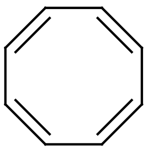
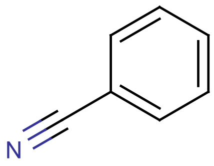
number of ...	
CIFs	6738
CIFs parsed to CML	6738
molecules extracted	6455
molecules suitable for calculation (original protocol)	1181
disordered molecules otherwise suitable for calculation	65
calculations (total)	1181
calculations failed (original protocol)	180
calculations failed (final protocol)	14

Table 8.1: The breakdown of the calculation statistics.

8.1.1 Insufficient Time

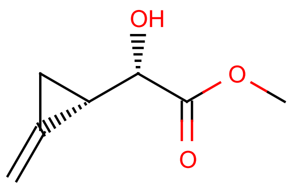
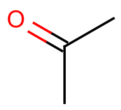
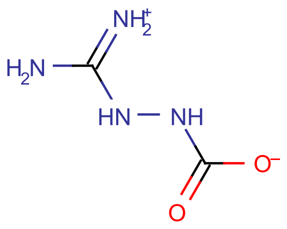
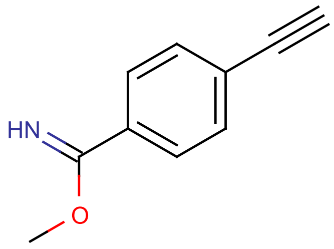
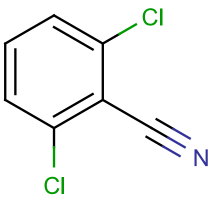
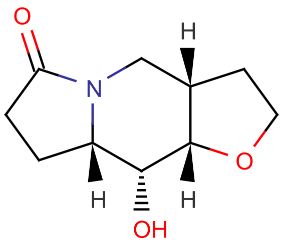
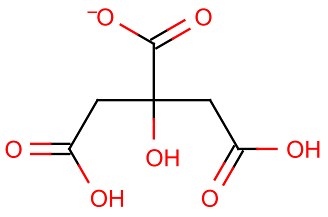
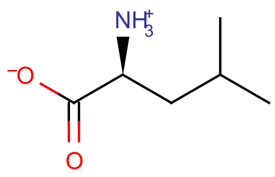
Table 8.2 shows the unique CTs of the molecules that did not have sufficient time to complete the minimisation process. The dataset of completed calculations was searched for the CTs of each of the molecules which failed. This search revealed cases where the calculation of the same CT (with very similar initial geometries) resulted in both failures and successfully completed computations. Figure 8.1 show an example of the energy profiles of the successful and failed calculations. The profiles suggest that there is a very small radius of convergence for the energy minimisation algorithm implemented in GAMESS. The 21 failures of this type represent a failure rate of 1.8%. This figure matches that found for the calculation of the MOPAC optimised structures which, taken in conjunction with the large spread of run times for a given number of non-hydrogen atoms, suggests that a failure rate of ca. 2% should be expected. There does not appear to be any way to differentiate the CTs giving rise to failures of this type from those which successfully complete in the time limit. Therefore no further refinement were made to the protocol.

Table 8.2: The molecules which did not have sufficient time to complete. The numbers below each molecule are: the number of instances of this molecule which did not have sufficient time to complete and, in brackets, the number of instances of the molecule which successfully completed in the time limit.

 1 (0)	 1 (0)
 1 (0)	 2 (32)
 1 (0)	 2 (2)
 1 (1)	 1 (0)

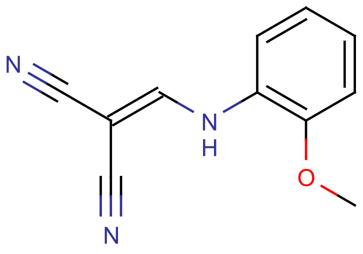
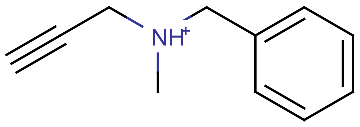
Continued on Next Page…

Table 8.2 – Continued

 1 (0)	 1 (9)
 1 (0)	 1 (0)
 1 (0)	 1 (0)
 1 (0)	 1 (1)

Continued on Next Page…

Table 8.2 – Continued

 1 (0)	 2 (2)

8.1.2 SCF Did Not Converge

The SCF failed to converge for 78 molecules under the original protocol. These 78 failures were caused by the 16 unique CTs shown in table 8.3. Again, the dataset was searched for instances where the same CT resulted in completed calculations; these are indicated in the table.

It was observed that the molecules for which the SCF did not converge all contain localised charges. However, the dataset included many molecules with localised charges for which the SCF did converge. It is therefore not reasonable to include a filter in the protocol to remove such molecules. The maximum number of iterations permitted for the SCF to converge is 30 by default in GAMESS. Increasing this value might allow more calculations to complete. However, these calculations are likely to require more time than is allowed by the protocol and therefore alteration was thought be be unnecessary.

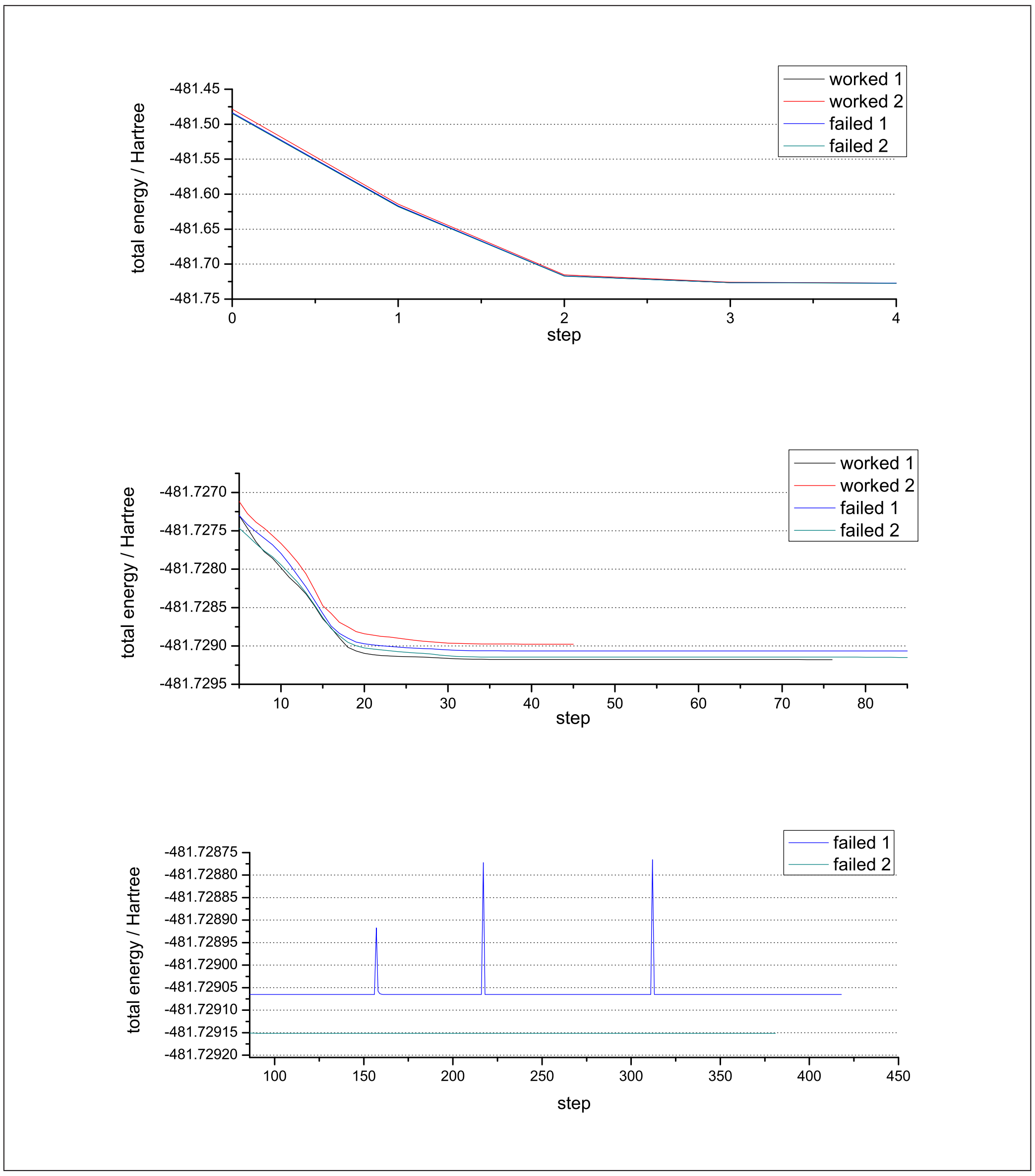
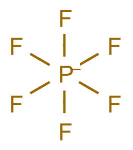
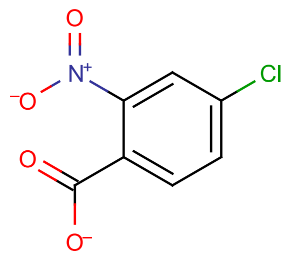
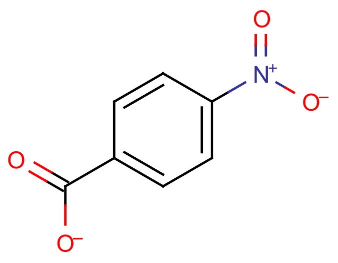
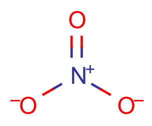
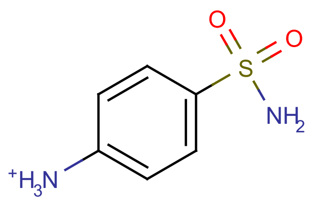
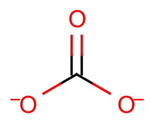
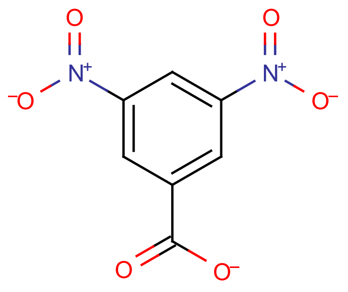
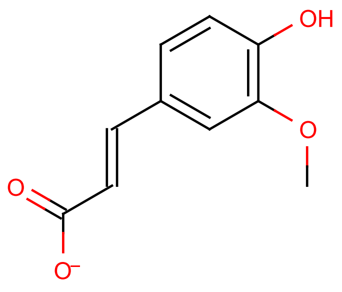


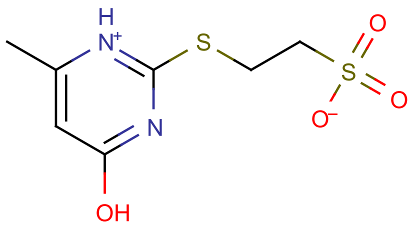
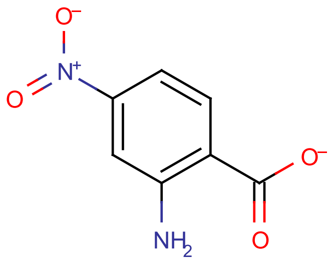
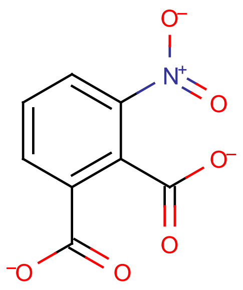
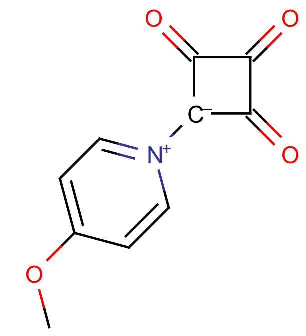
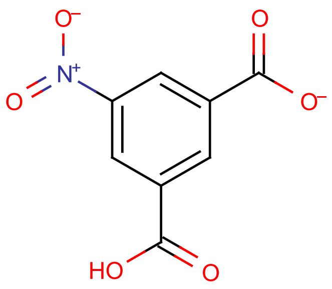
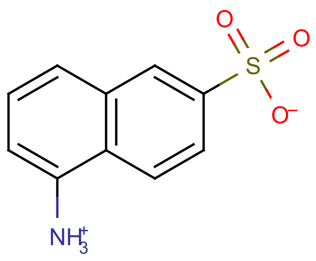
Figure 8.1: The energy profiles for four calculations on the same connection table (N-benzyl-N-methylprop-2-yn-1-aminium) with similar initial geometries. The y-axis scale is different for each graph.

Table 8.3: The molecules for which the SCF did not converge. The number of times each molecule was observed is indicated.

 10 (2)	 1 (0)
 7 (0)	 40 (0)
 1 (0)	 1 (0)
 6 (0)	 1 (0)

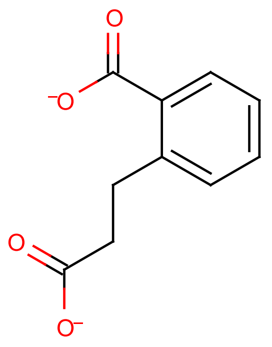
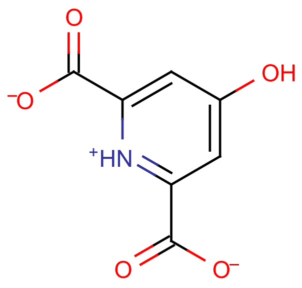
Continued on Next Page…

Table 8.3 – Continued

 1 (0)	 2 (0)
 2 (0)	 1 (0)
 1 (0)	 1 (0)

Continued on Next Page…

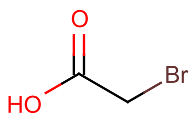
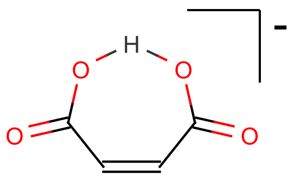
Table 8.3 – Continued

 2 (0)	 1 (0)

8.1.3 Bad Delocalised Coordinates Generated

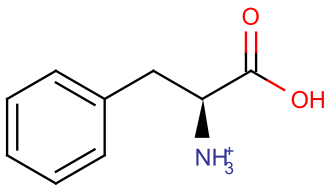
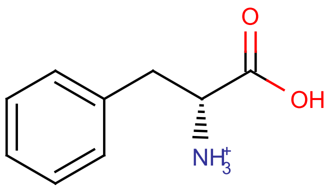
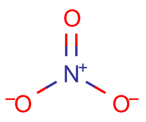
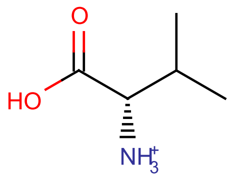
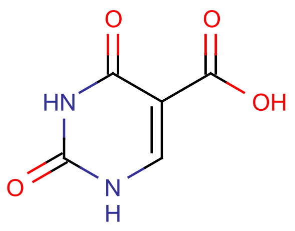
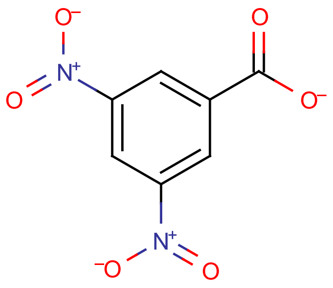
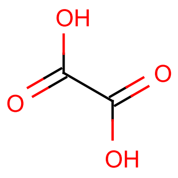
It was noted in section 5.3 that the automated creation of internal coordinates by GAMESS may fail. There were 16 instances of this failure from 9 unique CTs. These unique CTs are shown in table 8.4 — as before, the number of instances where the same connection table resulted in completed calculations is indicated in brackets. This is a previously reported GAMESS error and there is no easy way to determine which molecules it will effect. Therefore, no changes were made to the protocol.

Table 8.4: The molecules for which bad delocalised coordinates were generated. The number of times each molecule was observed is indicated.

 1 (0)	 3 (0)

Continued on Next Page...

Table 8.4 – Continued

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

8.1.4 Incorrect Charge or Multiplicity

There were found to be three reasons for this failure, all of which occurred in molecules suitable for calculation under the final protocol. The most common cause of this failure was the incorrect charge being specified in the CIF, or not being specified in a manner that was correctly parsable by the CIF2CML process. There is no way to account for author error. The checkCIF process does not report these as severe alerts although these mistakes were detected by the calculation process.

The CIF2CML process might be modified so that it places looser restrictions on the format of formulae that it can parse. This would allow the parsing of the specified formula even if it did not correspond to the format specified by the CIF dictionary [205]. However, the results of the OSCAR project suggest that it is impossible to correctly interpret loosely-defined data types with high precision. Failures of this type would therefore still be expected.

Two instances of this problem were caused by a mistake in the input file; one of the lines defining an atom in the file was longer than 80 characters. This causes GAMESS to skip the next input line which results in the next atom not being included in the calculation. The cause of the long lines is the use of the Java Double primitive to hold the coordinates. Whilst the CIF2CML program holds the fractional coordinates to the precision defined in the CIF, the cartesian coordinates created by the process are held to the maximum precision allowed by the Double primitive. Detection of this problem is possible before submission and was included in later protocols. However, it is still desirable that the underlying cause should be addressed; namely that the CIF2CML parser should ensure that the derived values should be given to the appropriate precision.

The final reason for this problem was incorrectly reported structures. For example, Newton *et al.* report the structure and formula of 2-C-hydroxymethyl-2,3-O-isopropylidene-D-ribono-1,5-lactam as $C_9H_{14}NO_5$ which matches the structure reported in the CIF [208]. However, this structure does not match the CT given in the article which is $C_9H_{15}NO_5$ (see figure 8.2).

Failures owing to the incorrect charge and/or multiplicity being specified accounted for 5.5% of the submitted calculations and were all caused by incorrect or inconsistent CIFs. Failures of this type take extremely little time to

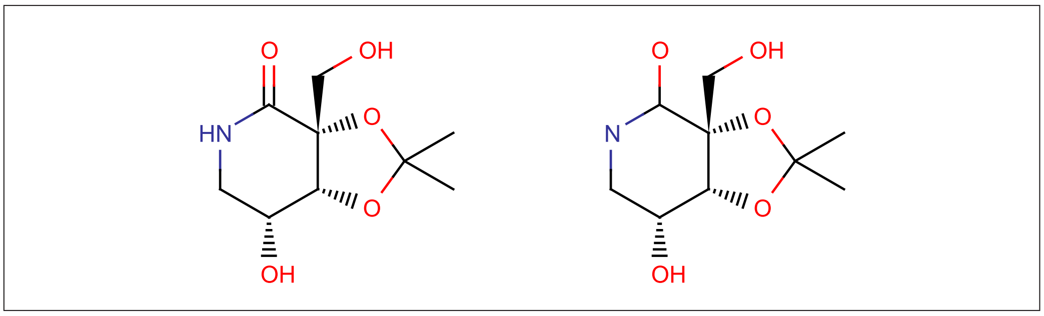


Figure 8.2: The CTs of 2-C-hydroxymethyl-2,3-O-isopropylidene-D-ribono-1,5-lactam as reported by Newton *et al.* The intended structure is shown of the left and the structure reported in the CIF on the right [208].

calculate (typically a few tenths of seconds) and therefore such computations could be used to automatically validate the reported crystal structures.

8.2 Proteus Molecules

Previous work highlighted the existence of molecules which when under going geometry optimisation showed a change in the CT (proteus molecules). The analytical tools used to determine differences in the geometry between the input and output structures require that there is no change of CT between the two. Therefore, before further processing, an InChI was generated for each molecule in each of the the parsed `data-and-coords.xml` files. The InChI was added as a child of the `molecule` element as a separately namespaced `identifier` element (see figure 8.3). A program was written to iterate through each of the InChIs and compare them to the InChI of the original (input) molecule. If the basic InChIs did not exactly match, a flag was added to the file to indicate that the molecule was protean.

Initial analysis of the proteus molecules presented some unexpected results; namely that although some of the molecules marked as protean did change CT at some point during the calculation, the input and output molecules had identical InChIs. To rectify this another flag was introduced

```
<scalar id='flag' dictRef='jat:semi_proteus_molecule' />
```

```xml
<cml xmlns:inchi="http://www.iupac.org/inchi" xmlns="http://www.xml-cml.org/schema">
  <molecule>
    <atomArray>
      ...
    </atomArray>
    <bondArray>
      ...
    </bondArray>
    <inchi:identifier>
      <inchi:basic>InChI=1/C6H11NO2/c8-6(9)5-3-1-2-4-7-5/h5,7H,1-4H2,(H,8,9)/t5-/m0/s1</inchi:basic>
      <inchi:auxinfo>
AuxInfo=1/1/N:11,14,8,17,6,20,3,1,2/E:(8,9)/it:im/rA:20OONHHCHCHHCHHCHHCHHC
/rB:...</inchi:auxinfo>
    </inchi:identifier>
  </molcule>
</cml>
```

Figure 8.3: An example of how a molecule's InChI was incorporated into the CML document.

These molecules showed a change of CT during the minimisation process but the optimised structure had an identical InChI to the input, whereas a true proteus molecule has a different InChI for the input and output structures.

28 proteus molecules were found, most of which were a result of the movement of a hydrogen atom to neutralise the charge on the molecule. A consequence of this was that most of the amino acids and derivative structures were removed from the dataset. These molecules tend to exist in the zwitterion form in the solid state and solution where they are able to make favourable ionic interactions. However, individual molecules (particularly *in vacuo*) are not usually stable in this form with respect to the neutral species.

Of the 28 proteus molecules 26 were as a result of charge reduction, in all but one instance this involved the movement of hydrogen atom(s). The other cause was the distance between a nitrogen and a boron atom increasing from 1.66Å to 1.84Å which was no longer considered to be within bonding distance by the analysis tools used. The calculated bond order reported by GAMESS for the N–B bond was 0.325, with overall charges of $+0.64$ and -0.77 on the nitrogen and boron atoms respectively. The two proteus molecules which were not caused by charge reduction were both found to be molecules that should have been charged but were not reported to be so in the CIF. Table H.1 shows an example of the geometries adopted during the calculation of a proteus molecule.

15 semi-proteus molecules were found. 13 of these were caused by 6- or 7-member ring formation between charged and uncharged carboxylic groups in 1,3 or 1,4 relative positions. A typical example of this can be seen in table H.2. The remaining two instances involved the distance between bonded atoms lengthening beyond the value considered to constitute a bond by the analysis software (see table H.3). Although the semi-proteus molecules are interesting, overall they do not effect the protocol because only the initial and final geometries are being compared.

8.3 CIF Analysis

To verify that the repository of CIFs used in this work was a representative subset of small molecule crystallography (as recorded in the CSD) a series of comparisons were drawn which are presented below. The statistics are derived from the 6675 CIFs which were parsed into CML to form the corpus for this work (set A) and the 775 CIFs which contained at least one molecule deemed suitable for calculation under the original protocol (set B).

The CCDC publishes yearly statistics, based on the structures in the the CSD as of the 1st of January each year. The statistics used below are taken from the 2007 publication. The derivation of the statistics from the datasets used in this thesis has been performed by automated processes wherever possible. This means that minor errors are interpreted as errors even if the error was trivially human-parsable.

Table 8.5 shows the analysis of crystal systems in the CSD and corpus of CIFs used in this thesis. The 6675 parsed CIFs contained 6673 instances where the crystal system was correctly defined. The two which did not, contained incorrect formatting and/or labels that were not defined in the CIF dictionary. One further result was excluded because it contains a spelling mistake in the definition (‘rombohedral’).

System	% of CSD	Entries set A	% of set A	Entries set B	% of set B
Triclinic	23.5	1543	23.1	160	20.6
Monoclinic	52.7	3690	55.3	439	56.6
Orthorhombic	18.9	1203	18.0	165	21.3
Tetragonal	2.2	108	1.6	7	0.9
Trigonal	1.7	59	0.9	4	0.5
Hexagonal	0.5	38	0.6	0	0
Cubic	0.4	29	0.4	0	0
Rhombohedral	n/a	2	0.0	0	0
Total	99.9	6672	100	775	100

Table 8.5: The cell symmetries for the CSD and the corpus for this work. The CSD results are obtained from the 397,293 CSD structures for which the space group is fully defined.

Table 8.6 shows the distribution of Hermann-Mauguin space groups for the datasets. The 6675 parsed CIFS yielded 6671 instances with the Hermann-Mauguin space group correctly defined. The four files for which the space group could not be identified used labels which were not defined in the CIF dictionary.

Table 8.7 shows the R-factor statistics for the datasets. The CIF dictionary allows several to define the R-factor, all of which are valid. This work has only considered the conventional R-factor defined by the CIF dictionary which is labelled as

_refine_ls_R_factor_gt

The required R-factor was contained in 6650 CIFs, the remaining 25 files had the R-factor given in an alternative format. The CSD contains 3.8% structures with unreported R-factors, although it should be noted that these are from short communications and most commonly from the earlier literature [204]. The corpus for this work contains 0.4% structures with the conventional R-factor missing, although a valid alternative form was always present.

Space group	% of CSD	Entries set A	% of set A	Entries set B	% of set B
P2(1)/c	35.1	1463	21.9	181	23.4
P2(1)/n	n/a	997	14.9	134	17.3
P2(1)/a	n/a	68	1.0	12	1.6
P-1	22.6	1498	22.5	151	19.5
P2(1)2(1)2(1)	8.1	458	6.9	83	10.7
C2/c	7.9	552	8.3	36	4.7
P2(1)	5.5	298	4.5	54	7.0
Pbca	3.5	257	3.9	37	4.8
Pna2(1)	1.4	95	1.4	21	2.7
Pnma	1.3	92	1.4	0	0
Cc	1.1	46	0.7	3	0.4
P1	1.0	43	0.6	8	1.0
Pbcn	0.9	41	0.6	4	0.5
C2	0.9	54	0.8	3	0.4
Pca2(1)	0.7	61	0.9	10	1.3
R-3	0.6	24	0.4	1	0.1
P2/c	0.6	14	0.2	0	0
P2(1)/m	0.6	34	0.5	0	0
C2/m	0.5	54	0.8	0	0
P2(1)2(1)2	0.4	18	0.3	3	0.4
Pc	0.4	16	0.2	5	0.6
Pccn	0.4	26	0.4	3	0.4
Fdd2	0.3	21	0.3	2	0.3
I4(1)/a	0.3	15	0.2	3	0.4
Total	94.1	6671	93.6	773	97.5

Table 8.6: Hermann-Mauguin space group statistics for the CSD and the test corpus. The CSD results are obtained from the 397,293 CSD structures for which the space group is fully defined, and represents $\geqslant 0.3\%$ of the structures.

R-factor	% of CSD	Entries set A	% of set A	Entries set B	% of set B
$R \leqslant 0.030$	9.2	940	14.1	86	11.1
$0.030 < R \leqslant 0.040$	19.8	1766	26.5	219	28.3
$0.040 < R \leqslant 0.050$	22.4	1890	28.3	246	31.7
$0.050 < R \leqslant 0.070$	27.9	1709	25.6	190	24.5
$0.070 < R \leqslant 0.090$	10.5	279	4.2	27	3.48
$0.090 < R \leqslant 0.100$	2.4	46	0.7	6	0.8
$0.100 < R \leqslant 0.150$	3.4	18	0.3	1	0.1
$0.150 < R$	0.7	2	0.0	0	0
not reported	3.8	25	0.4	0	0

Table 8.7: R-factor statistics for the CSD and the corpus for this thesis.

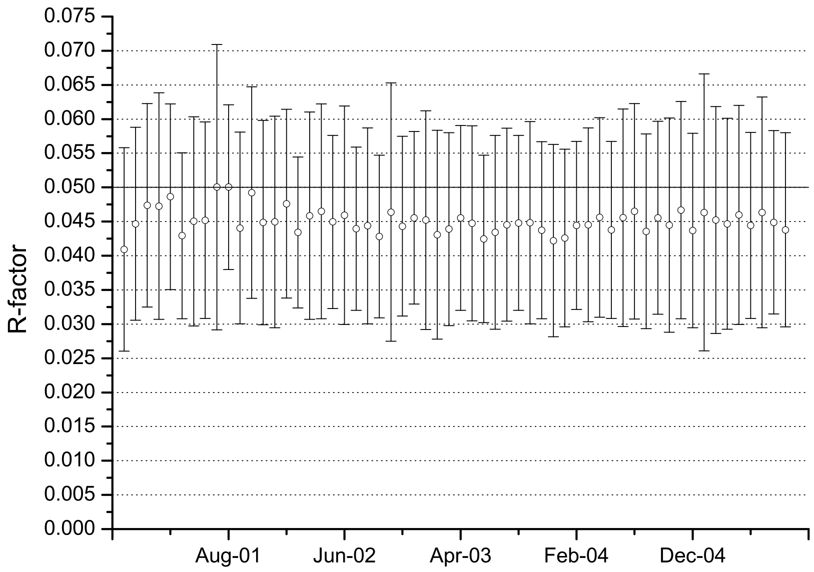


Figure 8.4: The mean R-factor of all the CIFs accepted by Acta Crystallographica Section E by month. The error bars show the standard deviation of the values.

Tables 8.5 and 8.6 suggest that the dataset used for this thesis does form a reasonable subset of small molecule crystallography. However, it was observed that the corpus of CIFs used in this study contained a greater percentage of lower R-factors than those in the CSD. This was expected because this study only considers CIFs published between the years 2000 and 2005 inclusive whereas the CSD contains much older structures since when there have been significant improvements in crystallographic experiments. Figure 8.4 shows the mean R-factor per month of all the CIFs accepted by Acta Crystallographica Section E for publication between November 2000 and August 2005. The data is derived from the 6123 CIFs used in this study that contained both the required R-factor and the journal acceptance date. The graph shows that both the mean and the standard deviation of the R-factor has remained approximately constant since November 2000. The graph also shows that the implementation of the R-factor $\leqslant 0.05$ filter in the protocol removes at most 50% of the structures published in a particular month.

8.4 Bond Lengths

Figure 8.5 shows a histogram of the reported estimated standard deviations (esds) for the 990 bonds which pass the final protocol. It is important to have an idea of the average esd to determine whether or not differences in bond length between those reported in the CIF and those calculated by GAMESS are significant. The average esd is 0.003Å and s=0.002Å, therefore it is likely that bond lengths determined by the two methods that differ by approximately 0.003Å may be explained by the expected spread of the data. The graph shows that most of the esds are less than 0.005Å, which might be a suitable filter to introduce in future protocols.

8.4.1 S–X Bonds

The comparison of the bond lengths in the previous chapter indicated that the GAMESS bond length was consistently longer than that reported in the CIF. This lengthening effect was most noticeable for longer bonds which were identified as S–X bonds (X=C,N). The seven molecules containing cyclic S–X

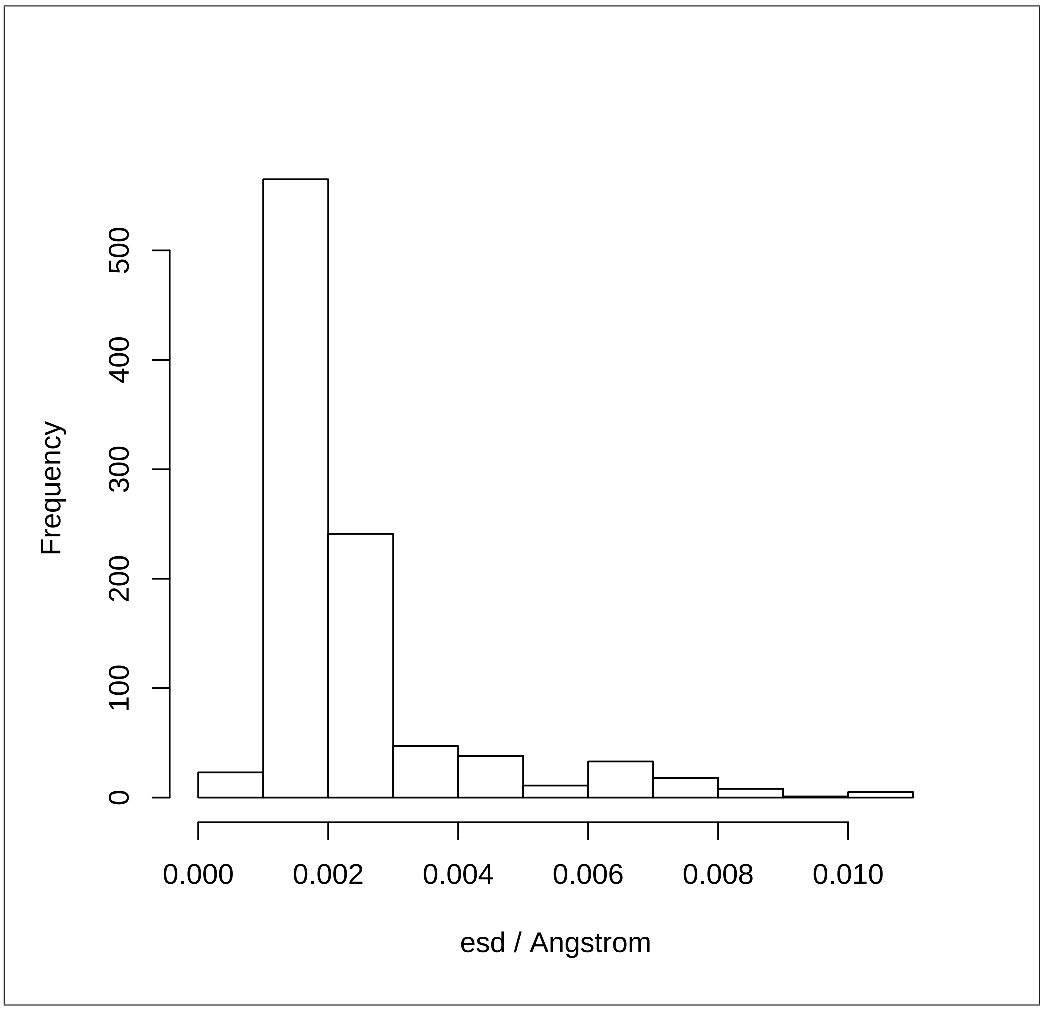


Figure 8.5: The reported esds of the 990 bonds which pass the final protocol, $\overline{esd}$=0.003Å, s=0.002Å.

molecule	GAMESS		*GAUSSIAN03*	
name	bond length / Å			
	a	b	a	b
lh6438molecule1	1.718	1.791	1.719	1.790
lh6379molecule1	1.742	1.732	1.730	1.741
ac6153molecule1	1.759	1.800	1.762	1.804
ob6428molecule1	1.771	1.769	1.766	1.767
bt6436molecule1	1.839	1.893	1.839	1.888
bt6436molecule2	1.838	1.893	1.839	1.890
cv6296molecule1	1.783	1.759	1.783	1.759

Table 8.8: The bond lengths of the S–X (X=C,N) bonds for the seven molecules shown in figure 8.6. Both calculations were performed using 6-31G*/B3LYP.

bonds in the dataset are shown in figure 8.6. There is no clear reason for the discrepancy in the bond lengths so it was thought that a higher level calculation may be required.

The molecules were submitted for calculation by GAMESS using a large basis set (6-311G**) and the B3LYP exchange function. Unfortunately these calculations did not complete in the time limit and therefore *GAUSSIAN03* was used for the calculations instead. The *GAUSSIAN03* program was a very recent addition to the computating facilities at the UCC and runs on the recently acquired Dove cluster. This is an Opteron-based compute cluster intended for parallel work. It has a head node with dual Opteron 246 CPUs and 4GB RAM. There are 8 compute nodes with dual, dual-core Opteron 265 CPUs and 4GB of RAM. This hardware is significantly faster than that of the Kellogg cluster which reduces the run times approximately eight-fold. This assumes that the implementation of the DFT code does not scale significantly differently in the two programs.

The *standard* 6-31G*/B3LYP calculation was repeated on *GAUSSIAN03* to verify that the two methods produced similar bond lengths for the molecules (see table 8.8). Three further calculations were performed on each molecule:

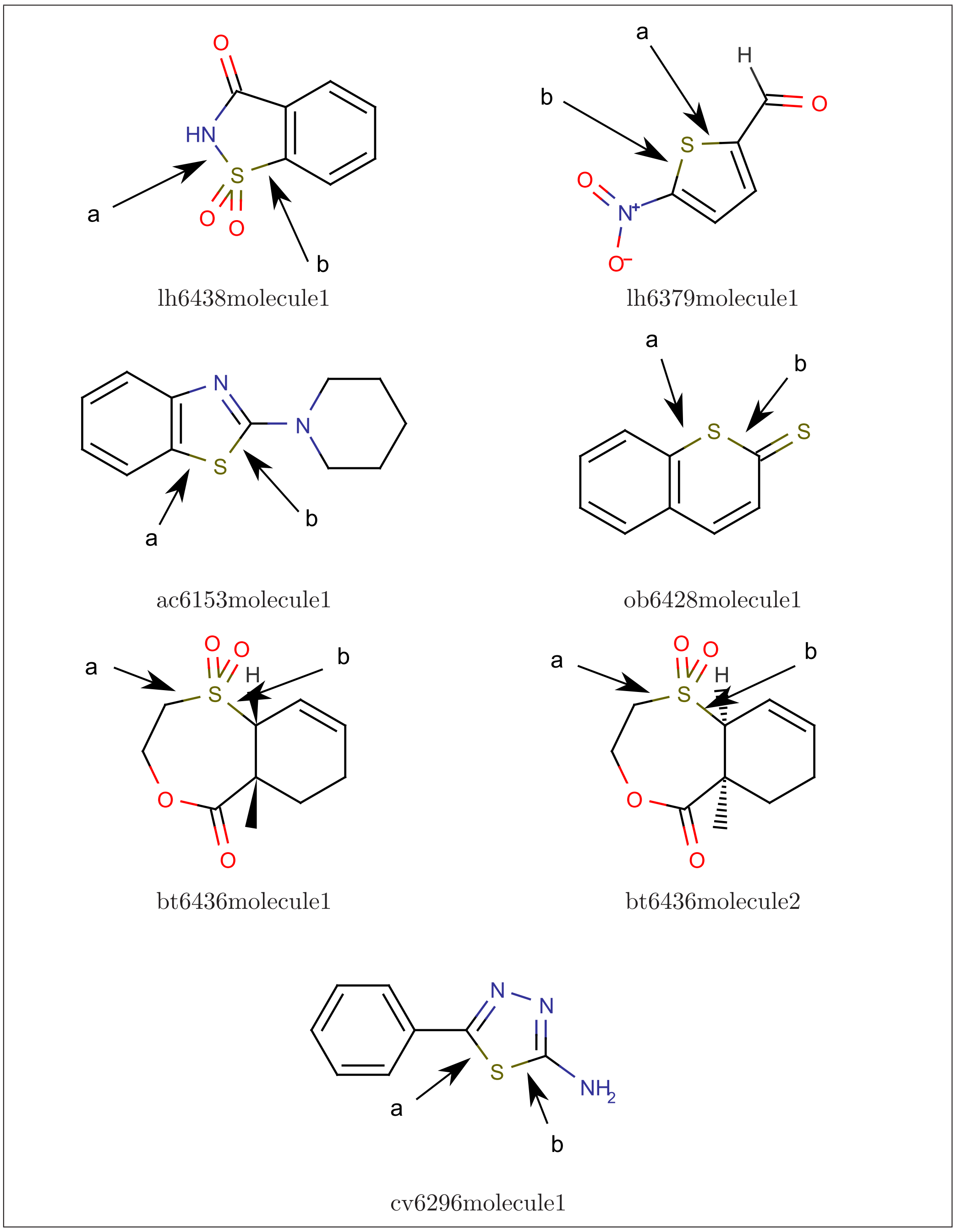


Figure 8.6: Long bonds to sulphur. The indicated bonds show a large difference between the calculated and CIF bond lengths.

	CIF		6-311G** B3LYP		6-31G* MP2		6-311G** MP2	
	a	b	a	b	a	b	a	b
lh6438molecule1	1.664	1.756	1.716	1.792	1.707	1.772	1.701	1.772
lh6379molecule1	1.720	1.709	1.782	1.739	1.713	1.726	1.709	1.721
ac6153molecule1	1.739	1.771	1.762	1.802	1.744	1.781	—	—
ob6428molecule1	1.744	1.730	1.763	1.766	1.752	1.754	1.749	1.750
bt6436molecule1	1.788	1.825	1.840	1.893	1.810	1.843	—	—
bt6436molecule2	1.789	1.826	1.840	1.894	1.810	1.843	—	—
cv6296molecule1	1.752	1.749	1.782	1.759	1.750	1.739	1.743	1.734

all values are given in Å to four significant figures

Table 8.9: The S–X (X=C,N) bond lengths as reported in the CIF and calculated at various levels and methods of theory. The labels a and b indicate the bond referred to (see figure 8.6). Where no bond length is shown, the calculation failed to complete in the required time limit.

6-311G**/B3LYP, 6-31G*/MP2 and 6-311G**/MP2. The results of these calculations are shown in table 8.9. It was observed that simply increasing the size of the basis set did not improve the agreement with the experimental values. However, the calculations at a higher level of theory did show a general improvement. This suggests that 6-31G*/B3LYP is not sufficient for accurate calculations involving second row elements. Unfortunately the time required for calculations at higher levels of theory is often prohibitive. The protocol must therefore be modified to only include bonds involving first row elements.

8.4.2 All Bonds

To this point, all the QQ graphs have been plotted using the same axes limits to make it easier to detect the effect of implementing the various filters. The limits were chosen so that all outliers would be visible and were $-0.25 \leqslant \Delta R$ / Å $\leqslant 0.25$ on the y-axis and ± four standard deviations of the standard normal distribution on the x-axis. However, now that the major outliers have been identified and removed for justifiable reasons, it is instructive to examine the distributions in more detail.

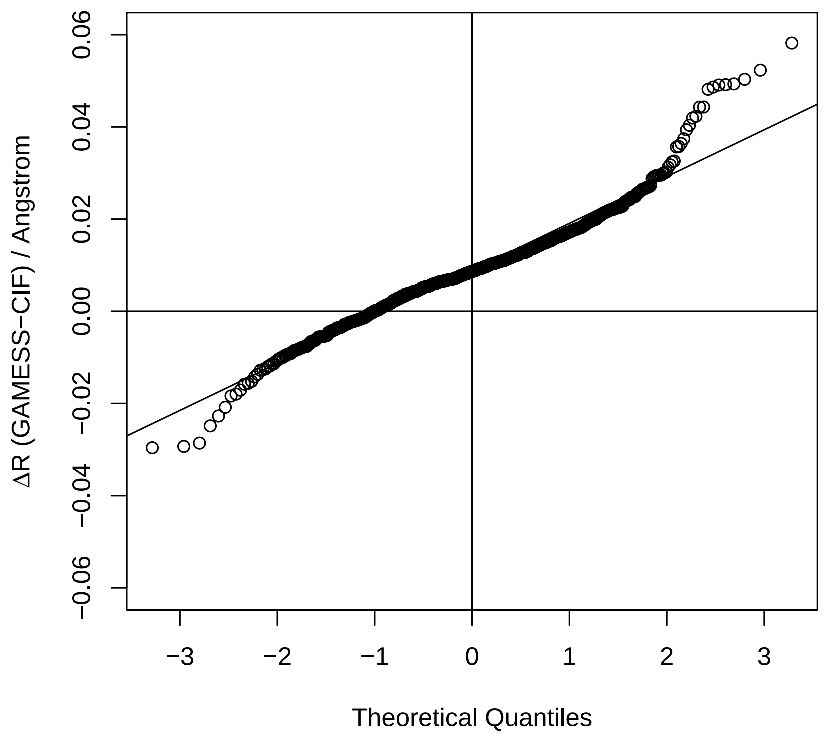


Figure 8.7: QQ plot of the 976 bonds in the dataset after the imposition of all the filters. The data is mostly normally distributed with $\overline{\Delta R}$=0.009Å and s=0.010Å although unusual behaviour is still observed at both long and short bond lengths.

The Shapiro-Wilk W test [207] tests the null hypothesis that a sample $x_1, \ldots, x_n$ are from a normally distributed population. A W statistic of 1 is found for data that is perfectly normally distributed. The test also provides a p-value which is used to assess whether or not the observed deviation from normality is significant. For example, a W statistic of 0.97 with p-value of 0.05 suggests that there is no evidence to reject the null hypothesis at a 95% confidence level. A lower p-value would indicate that the W value is too extreme to be explained by chance variation (*i.e.* evidence against normal distribution) and the null hypothesis should be rejected.

Figure 8.7 shows the QQ plot of all the bonds in the dataset after the imposition of all the filters. It is observed that although the data is mostly normally distributed there are significant outliers at both ends of the distribution. The Shapiro-Wilk W test gives W = 0.960, p-value = 1.04×10^{-15}, the null hypothesis is therefore rejected, although the W value near unity suggests that the data is almost normally distributed. The deviation from normality is likely to be caused by the tails of the distribution which are examined in the next sections.

8.4.3 C–C Bonds

Figure 8.8 shows the 752 C–C cyclic bonds in the dataset after the imposition of all the filters. The molecules giving rise to those bonds with $\Delta R > 0.03$Å were examined and all found to be involved in hydrogen bonds in the crystal that had subsequently been lost in the calculation. An example of this effect is shown in figure 8.9. As expected the C–O bond length decreases when hydrogen bonding is lost whilst the adjacent C–C bond lengths both increase. The W value was 0.928 and p-value $< 2.2 \times 10^{-16}$, the null hypothesis is therefore rejected although the W value indicates that the data is nearly normally distributed. The tails of the distribution are the likely cause of this deviation from normality and may be explained by the loss of crystal packing effects when the calculation is performed.

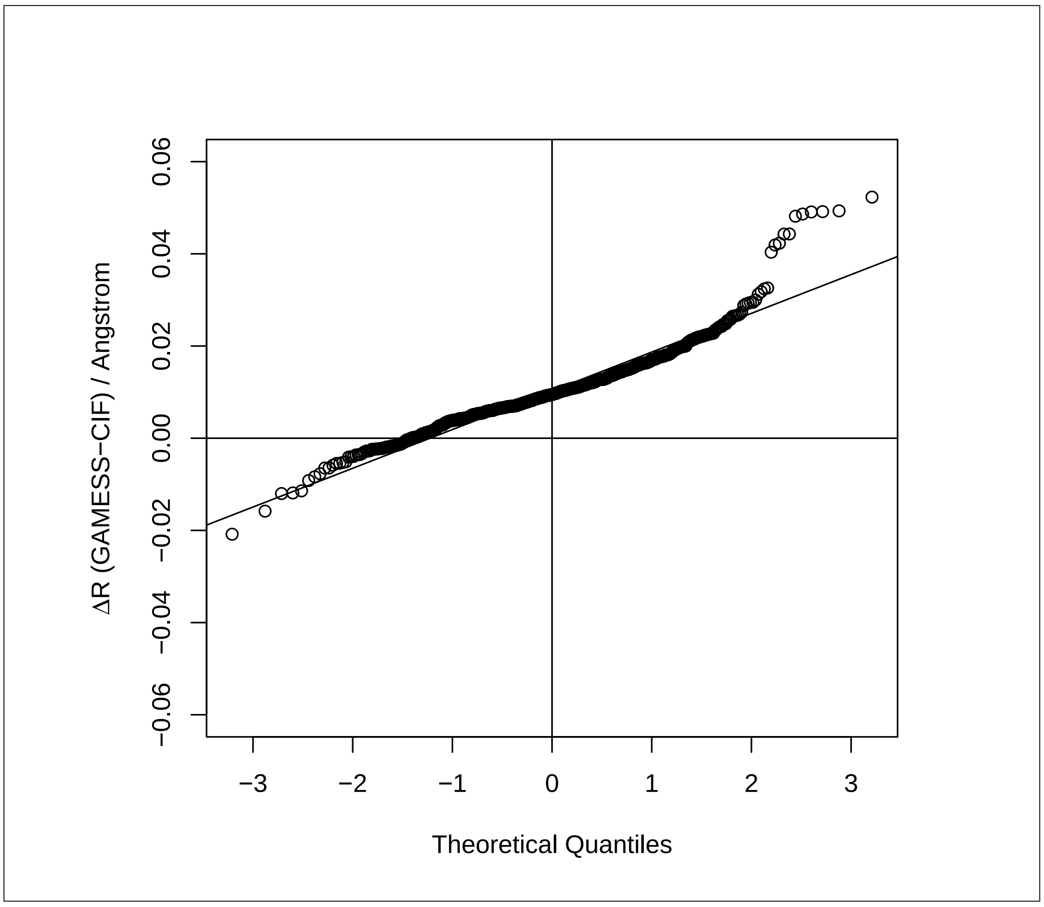


Figure 8.8: QQ plot of the 752 C–C cyclic bonds in the dataset after the imposition of the final protocol. The data is mostly normally distributed with $\overline{\Delta R}$=0.010Å and s=0.008Å. There appears to be a discontinuity at $\Delta R > 0.03\mathring{A}$.

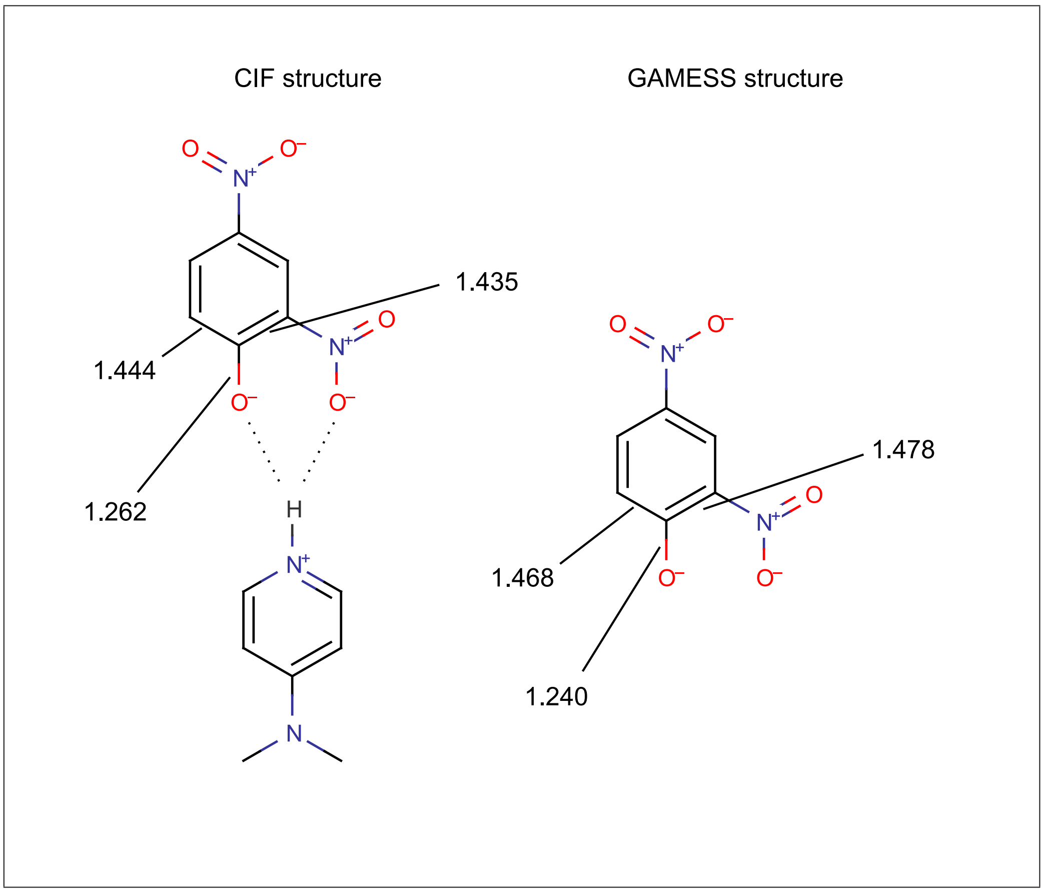


Figure 8.9: The loss of hydrogen bonding can effect the bond length of cyclic bonds. The bond lengths indicated are in Å.

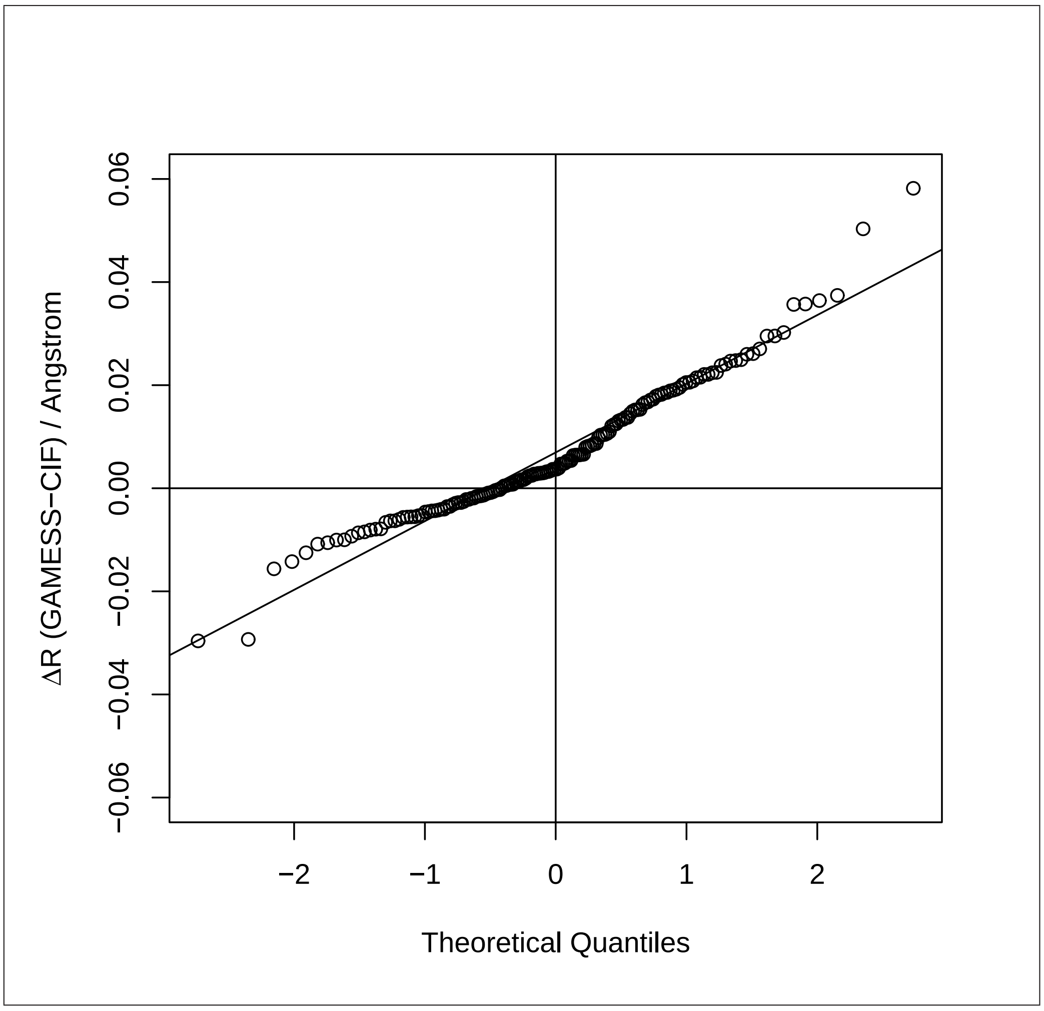


Figure 8.10: QQ plot of the 160 C–N cyclic bonds in the dataset after the imposition of the final protocol. The data appears normally distributed with $\overline{\Delta R}$=0.007Å and s=0.013Å.

8.4.4 C–N and C–O Bonds

There are 160 C–N bonds and 54 C–O bonds in the dataset after all the filters are imposed. The QQ plots of these are shown in figures 8.10 and 8.11 respectively. Both bond types show approximately normally distributed data. The Shapiro-Wilk W tests for the C–N and C–O ΔR were W = 0.962, p-value = 2.00×10^{-4} and W = 0.953, p-value = 3.23×10^{-2} respectively. Again the outliers are caused by the loss of crystal packing effects on calculation. These tests indicate that the data is not normally distributed, but is almost so. There are no other bond types with more than 13 instances in the dataset.

8.4.5 PLATON

Rigid-body model libration corrections can be calculated using PLATON [209]. Unfortunately, these calculations cannot easily be automated because the program cannot be run entirely from the command line. However, to investigate the effect of the possible corrections, the program was run manually on the 129 molecules which contained cyclic bonds that passed the entire protocol. This dataset comprised 990 bonds but PLATON was unable to calculate libration corrections for 30 of these bonds. In all cases the PLATON corrected bond length was longer than that reported in the CIF (the minimum correction was 0.001Å and the maximum 0.014Å), $\overline{\Delta R}$(PLATON-CIF)=0.003Å with s=0.002Å. The X–S bonds (X=C,N) were lengthened by an average of 0.003Å which improved the agreement with the calculated values for these bonds. $\overline{\Delta R}$(GAMESS-PLATON)=0.006Å with s =0.010Å, the maximum and minimum ΔR(GAMESS−PLATON) found were 0.064Å and −0.031Å respectively. These results suggest that the libration corrected bond lengths agree well with the calculated bond lengths although the spread of the data is significantly larger than the $\overline{esd}$ of the bond lengths. Much of this variation is expected to be a result of the loss of crystal packing effects when the calculation is performed on isolated molecules *in vacuo*.

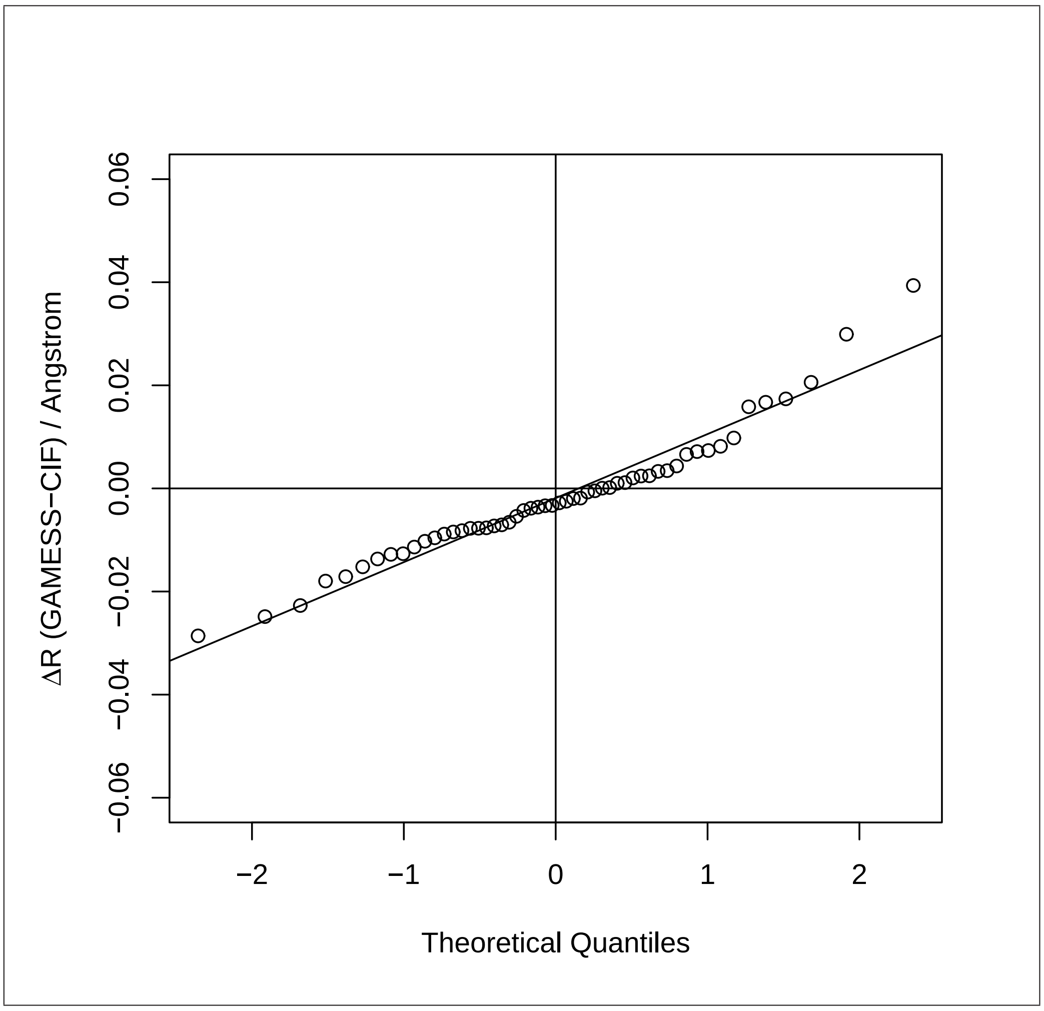


Figure 8.11: QQ plot of the 54 C–O cyclic bonds in the dataset after the imposition of the final protocol. The data appears normally distributed with $\overline{\Delta R}$=−0.002Å and s=0.012Å.

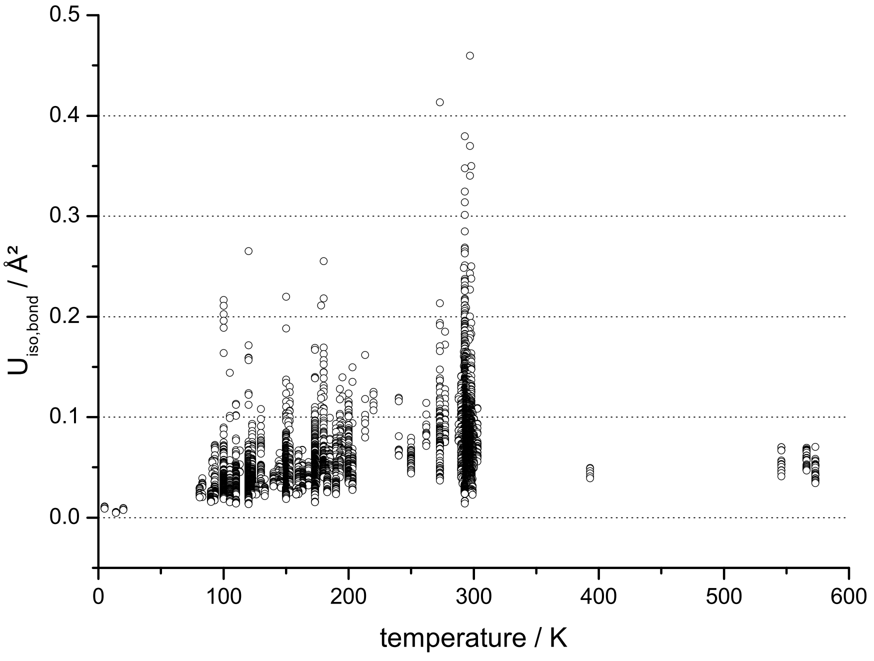


Figure 8.12: The $U_{iso,bond}$ for the 9512 bonds (excluding those to hydrogen) calculated by GAMESS under the original protocol plotted against the temperature at which the experiment was conducted.

8.5 $U_{iso,bond}$

The

```
_atom_site_U_iso_or_equiv
```

data item is used to report the equivalent isotropic parameter U_{eq} for each atom in a CIF. The parameter is expected to depend on temperature. The $U_{iso,bond}$ between two bonded atoms i and j was defined as:

$$U_{iso,bond} = \sqrt{U_{eq,i}^2 + U_{eq,j}^2} \tag{8.1}$$

where $U_{eq,i}$ is the reported U_{eq} for atom i. Figure 8.12 shows $U_{iso,bond}$ plotted against temperature for all 9512 bonds (excluding those to hydrogen) that were calculated by GAMESS under the initial protocol. There are several features of interest; firstly there are structures which have reportedly been determined at 573, 566, 546 and 393 K. These are not reported as high temperature studies in the literature and look far more reasonable when 273 K is subtracted from the values. It is extremely likely that the authors had incorrectly reported the temperature in Celsius rather than of Kelvin. The $U_{iso,bond}$ appears to be only weakly correlated with temperature ($\rho = 0.45$).

The bonds giving rise to some of the highest values of $U_{iso,bond}$ were examined and were found to be solvent molecules. An example of a structure giving rise to a high $U_{iso,bond}$ is shown in figure 8.13. The extreme eccentricity of the thermal displacement ellipsoids in the solvent molecule suggests that there is likely to be unreported disorder (minor conformations) present.

Figure 8.14 shows $U_{iso,bond}$ against temperature for the bonds which pass the final protocol (although both cyclic and acyclic bonds are permitted). It is observed that there is still only weak correlation ($\rho = 0.48$) between temperature and $U_{iso,bond}$. The largest values of $U_{iso,bond}$ were from the C–F bonds in 2,2,2-trifluroethanol molecules. These molecules should have been removed as solvents but were missed. The list of solvent/ion/guest molecules was compiled by hand and is therefore likely to have some omissions. The

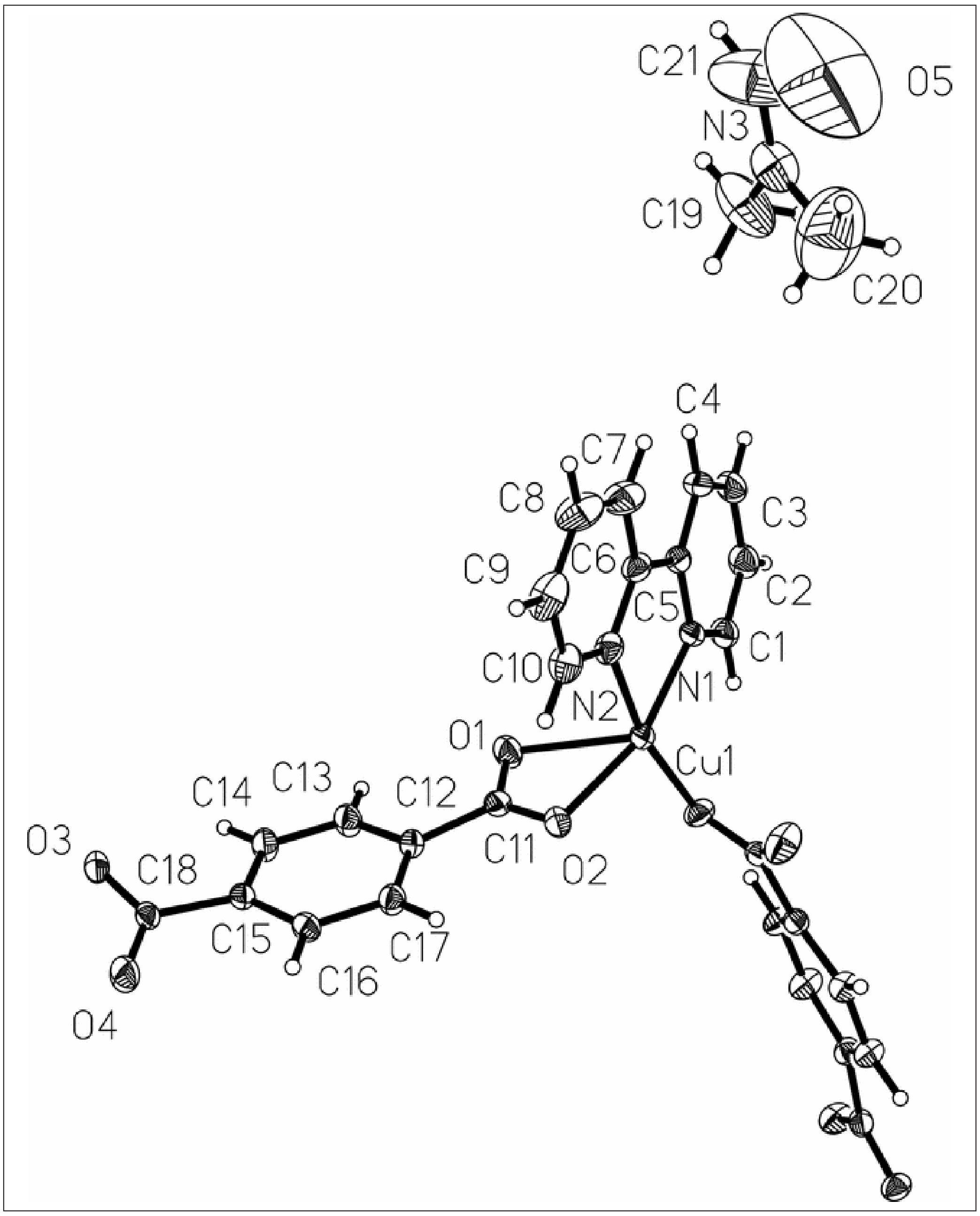


Figure 8.13: Displacement ellipsoids at the 50% probability level showing extremely large thermal motion for the *N*,*N*-dimethylformamide molecule. The size and extreme eccentricity of the O5 ellipsoid in particular suggests that minor conformations may be present. The figure has been taken from the article by Li and Xiao [206].

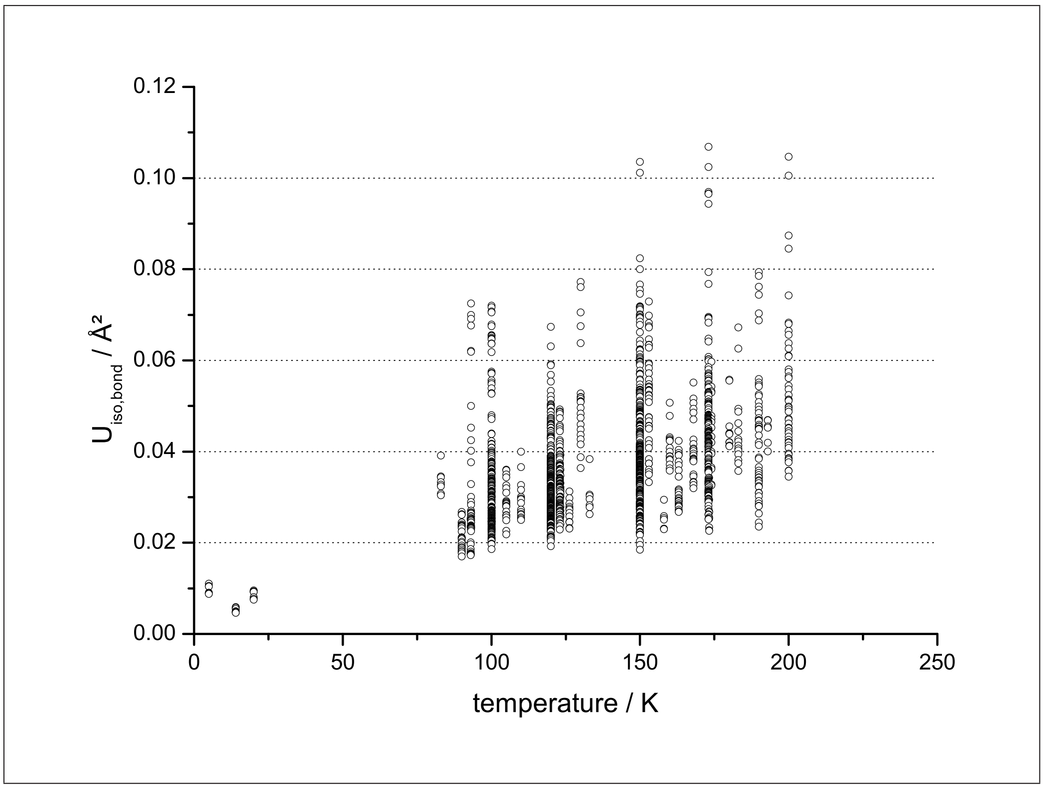


Figure 8.14: The $U_{iso,bond}$ for the 990 bonds (excluding those to hydrogen) that pass the final protocol plotted against the temperature at which the experiment was conducted.

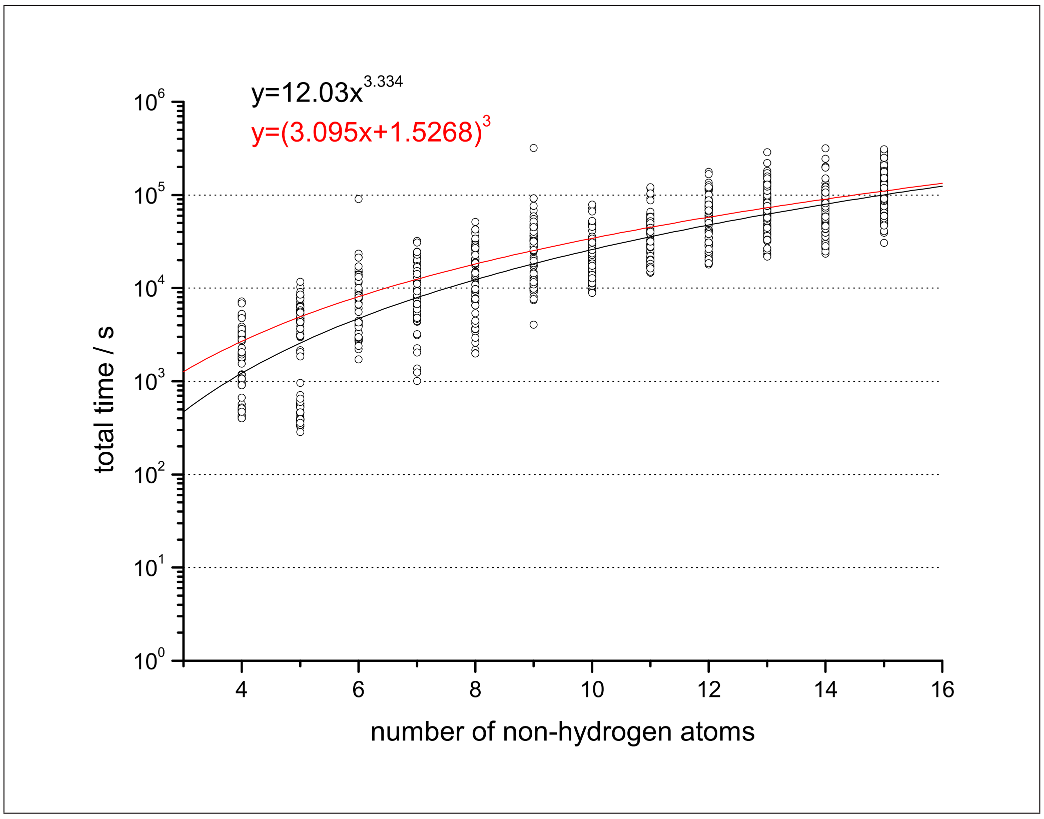


Figure 8.15: The total calculation time for the geometry optimisation of structures taken from crystallography scales less favourably than the predicted cubic dependence (red line). However, the predicted run time consistently over estimates the actual value for $n < 15$.

list of molecules currently identified as solvent/ion/guests is given in full in appendix F. Figure 8.14 suggests that a further filter might be added to the protocol, namely that bonds with $U_{iso,bond} \geqslant 0.1$ Å^2 should be omitted from analysis. Unfortunately $U_{iso,bond}$ is a derived quantity and is not reported in the CIF which makes the filter more difficult to implement.

8.6 Time

Figure 8.15 shows that the total run times for the geometry optimisation of the structures obtained from crystallography scale less favourably ($n^{3.334}$)

number of non-H atoms	mean total time / s	standard deviation / s	predicted mean total time / s	predicted standard deviation / s
4	2100	1600	2700	980
5	3500	2900	4900	2400
6	9300	12000	8100	4400
7	10000	8100	12000	7300
8	16000	11000	18000	11000
9	32000	35000	25000	15000
10	25000	15000	34000	21000
11	40000	23000	45000	28000
12	59000	35000	58000	36000
13	80000	44000	73000	45000
14	76000	48000	90000	56000
15	110000	61000	110000	84000

all values given to 2 significant figures

Table 8.10: The observed run times are reasonably well predicted using equation 5.1. The largest observed deviation occurs for molecules containing 14 non-hydrogen atoms with the predicted average run time overestimating the true average run time by almost four hours. The predicted standard deviation consistently underestimated the observed values.

than those that were already optimised using MOPAC which scale as $n^{2.9469}$. It is observed that there is still a large variation of run times for a particular number of non-hydrogen atoms. Table 8.10 shows that run times predicted using equation 5.1 are reasonable, the largest difference between the predicted and actual average run times being approximately 4 hours.

These comparisons support the theory that it is possible to predict the average run times for calculations using simple models, although the standard deviation was consistently underestimated. It is important to note that attempting to predict the time required for a single calculation is nonsensical given the extremely large standard deviations observed and that very similar starting structures may require significantly different calculation times (see figure 8.1). However, the upper bound on the time to calculate a sufficiently large dataset may be predicted with reasonable confidence. It is suggested that such a dataset should contain at least 100 molecules.

8.7 Applying the Protocol

Figure 8.16 shows the final protocol developed with each of the filters colour coded to indicate the reason for the implementation. There are five reasons;

Crystallographic effects These filters remove many of the poorly determined experimental values and are mostly based on data items reported in the CIF — the exception is the removal of solvent/ion/guest molecules.

GAMESS / time limitation These filters are required for the calculation to be completed within the time limit or for it to give meaningful results. For example, there is no point in attempting to compute the properties of atoms which are not well-described by the basis set being used.

Methodology The calculation is performed on single molecules *in vacuo* which removes many of the short contacts that are made in the crystalline form. These filters remove bonds which are likely to be affected by these disparities.

Author error Such errors are unfortunately unavoidable, although the use of validation tools before publication should reduce these in future, they will be present in the legacy literature.

X–Y (Y=Si,P,S,Cl,Br) The basis set and level of theory chosen (6-31G*/B3LYP) does not appear to be able to accurately calculate the bond lengths of bonds involving second row (or heavier) elements. Increasing the level of theory, to MP2 for example, would allow the calculation of heavier elements, but such calculations require much longer run times.

The identification of particular filters that relate to crystallographic effects allows the literature to be searched for high-quality structures (*i.e.* those

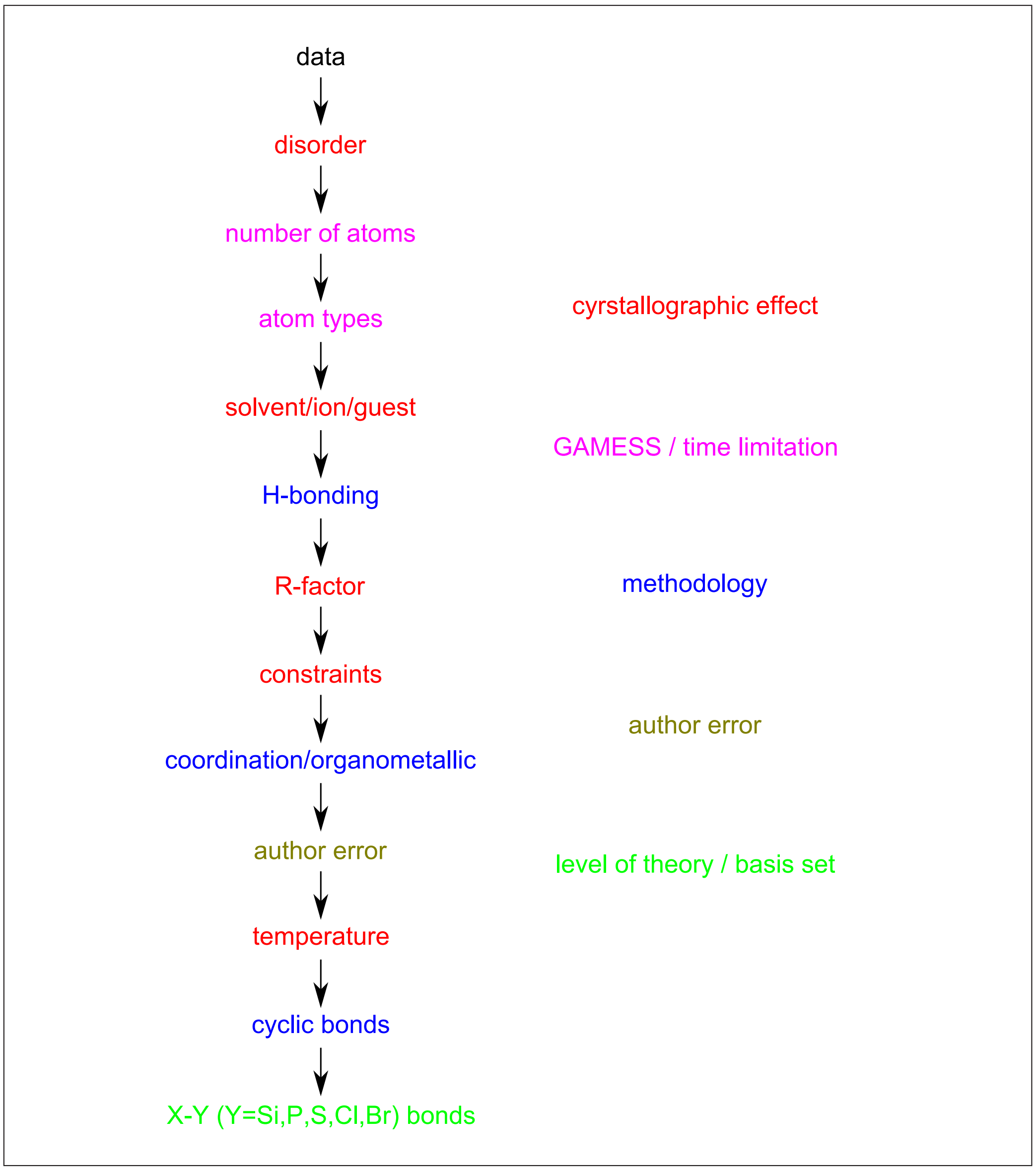


Figure 8.16: The filters imposed on the data colour coded by reason.

Filter	% pass
R-factor $\leqslant$ 0.05	66.5
Not solvent/ion/guest	89.4
No non-H refinement	83.0
Temperature $\leqslant$ 200K	32.9
No disorder	92.8
Pass all filters	18.0

Table 8.11: The percentage of the molecules which pass each individual crystallographic filter and the percentage which pass all the filters.

that pass all the crystallographic effect filters). The percentage of molecules extracted from the CIFs which pass each crystallographic filter and all the filters is shown in table 8.11.

Using the $\overline{esd}$ of bond lengths it is possible to estimate the maximum torsion angle in a toluene molecule that may be accounted for by random error in the coordinates as 0.85° (see figure 8.17). This agrees well with the $\overline{esd}$ reported for the torsion angles in the CIF which is of the order of 0.5°. The 6455 molecules obtained from the CIFs were searched for mono-substituted phenyl rings (the substituent being a carbon atom) contained in molecules that pass all the crystallographic filters of the final protocol. The largest deviation from planarity was found to be 4.1° for the cyclic atoms and 7.5° between the external carbon and the ring.

GAUSSIAN03 allows the calculation of optimised geometries with specified constraints. These constraints allow the torsion angle between particular atoms to be fixed at a particular value. Figure 8.18 shows the chosen constraints places on the torsion angles of toluene molecules. The energies of the geometry optimised structures relative to the completely flat molecule were found to be 0.35kJ mol^{-1} and 2.25kJ mol^{-1} for torsion angles t1=t2=175° and t1=t2=170° respectively. These calculations suggest that the energy required to produce the torsion angle of 7.5° is approximately 2kJ mol^{-1}. However, there is no simple interaction apparent in the crystal structure to

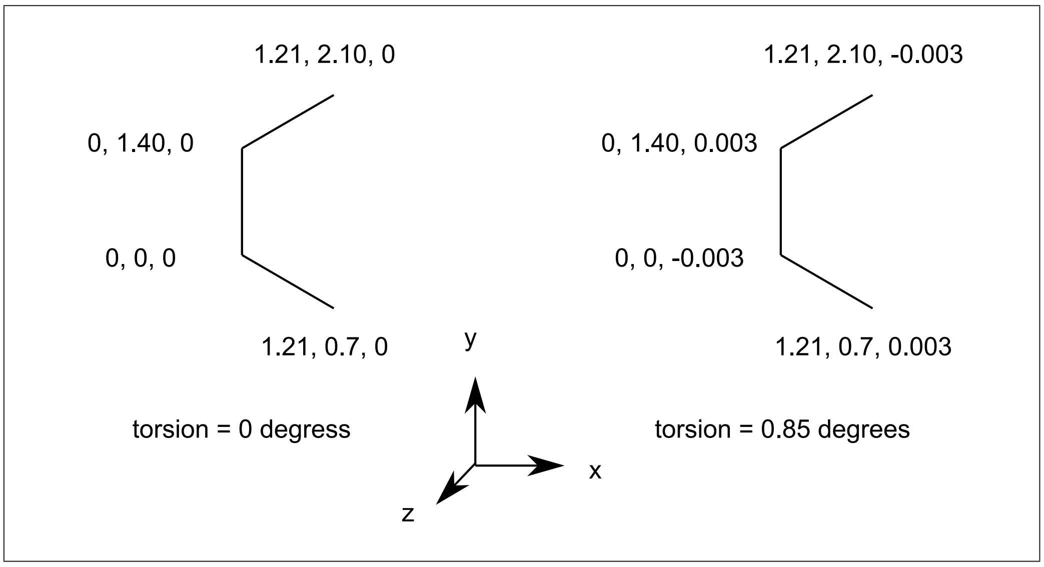


Figure 8.17: The maximum torsion angle of a phenyl ring that can be accounted for by the average esd in the atomic coordinates and a bond length of 1.4Å is 0.85°.

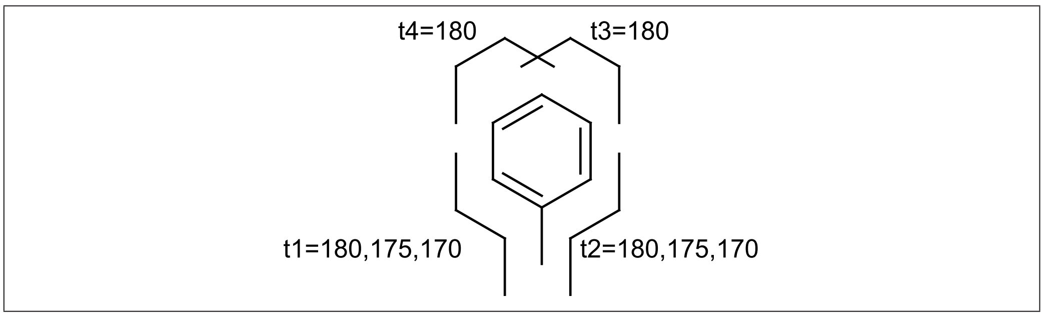


Figure 8.18: The torsion angles which were constrained during the geometry optimisation of toluene. The values shown are in degrees.

explain this distortion from planarity.

8.8 Conclusions

The information in CIFs can be parsed to more generally machine-understandable formats with extremely high recall and precision. This data includes the connection tables of the molecules and some derived data. This allows molecule-based data-drive science to be performed. The use of validation tools such as checkCIF before publication means that there are few errors in the data.

The comparison of the reported bond lengths to those calculated by GAMESS suggests that variations in bond length less than 0.03Å are the result of random error. The agreement between the values is improved by using rigid-body libration corrected bond lengths. Differences of bond lengths greater than 0.03Å may be explained by identifiable effects in general, but may merit further examination. The results have shown that the poor agreement between the bond lengths calculated by MOPAC and GAMESS for aromatic nitrogen-bearing moieties are likely to be caused by MOPAC.

The protocol developed allowed the identification of high-quality crystallographic structures (reducing recall at the expense of precision). Day is conducting further work which implements this protocol and involves inorganic structures [210]. The bond lengths in these high-quality structures, which account for ca. 20% of the recently reported structures, have an $\overline{esd}$ of 0.003Å. These structures form a dataset that can be used for the identification of *interesting* structures which can be reused to form the basis of future experiments.

Appendix A

Computational Chemistry

A.1 *ab initio* Calculations

Using atomic units, the time-independent molecular Schrödinger Hamiltonian is (ignoring all relativistic terms)

$$H = -\frac{1}{2}\sum_i \nabla_i^2 - \sum_{iA} \frac{1}{|\mathbf{r}_i - \mathbf{R}_A|} + \sum_{i>j} \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} + \sum_{A>B} \frac{Z_A Z_B}{|\mathbf{R}_A - \mathbf{R}_B|} \tag{A.1}$$

where i, j denote electrons at $\mathbf{r}_i, \mathbf{r}_j$ and A, B denote nuclei with charges Z_A, Z_B at $\mathbf{R}_A, \mathbf{R}_B$. Solutions (energies and wavefunctions) of the Schrödinger equation are obtained from

$$H\Psi = E\Psi \tag{A.2}$$

for fixed positions of the nuclei. $E \equiv E(\mathbf{R})$ is therefore the potential energy surface. The fundamental expansion functions used to find approximate solutions of Schrödinger's equation are Slater determinants, which have the form

$$\Psi = \frac{1}{\sqrt{n!}} \begin{vmatrix} \phi_1(1) & \phi_1(2) & \cdots & \phi_1(n) \\ \phi_2(1) & \phi_2(2) & \cdots & \phi_2(1) \\ \vdots & \vdots & & \\ \phi_n(1) & \phi_n(2) & \cdots & \phi_n(n) \end{vmatrix} \tag{A.3}$$

$$= \mathcal{A}(\phi_1\phi_2\phi_3\cdots\phi_n) \tag{A.4}$$

$$\mathcal{A} = \frac{1}{\sqrt{n!}}\sum_u^{n!} \sigma_u P_u \tag{A.5}$$

where P_u is a permutation of the coordinates in $\phi_1\phi_2\phi_3\cdots\phi_n$. A permutation is even ($\sigma_u = +1$) or odd ($\sigma_u = -1$) if it is made up of an even number or odd number of single interchanges. These determinants obey the Pauli principle.

Ideally hydrogenic type functions

$$R_{nl}(r)exp(-Z_nR)Y_{lm}(\sigma\phi) \tag{A.6}$$

would be used as expansion function for molecular orbitals. These can also be written as

$$r^n x^p y^q z^s exp(-Z_n r) \tag{A.7}$$

from which it is seen that such a set includes *s*, *p*, *d*, *f*, *etc.* atomic orbitals. However, it is impossible to (non-numerically) evaluate the one and two electron integrals which arise in the evaluation of matrix elements if such functions are used. These functions are usually referred to as Slater functions, or Slater Type Orbitals (STOs).

Boys [211] suggested using Gaussian basis functions, or Gaussian Type Orbitals (GTOs)

$$x^p y^q z^s exp(-ar^2) \tag{A.8}$$

with p, q, s integers, and $r^2 = x^2 + y^2 + z^2$. The angular parts of these functions are the same as the STOs but the radial part is different. The derivative of an s Gaussian is zero at the origin, unlike the STO. The Gaussian dies off with exponential quadratic dependence compared to the STO's linear dependence for large r. Thus GTOs have a totally different behavior to STOs at both small r and large r. However the key advantage of GTOs is that all the required integrals are easy to calculate. This follows from the fact that the product of two Gaussians is another Gaussian.

To overcome the less desirable short and long range behavior of Gaussians, it is common to used fixed combinations of one to six Gaussians as basis functions, chosen to make the combination look more like STOs. STO-3G for example means the use of a contracted combination of three Gaussians to represent a Slater function. Computational chemistry programs have developed a specific notation for basis sets of this sort often called Pople's basis set. The notation of the basis set is in the form N-ijG or N-ijkG where N is the number of Gaussian primitives (GTOs) for the inner shells, ij or ijk are the numbers of Gaussian primitives for contractions in the valence shell. N-ijG* denotes a polarized basis set augmented with d type functions on heavy atoms only, whilst N-ijG** or N-ijG(d,p) specifies a basis set with p-functions on hydrogen atoms as well.

A.1.1 Closed Shell Self Consistent Field Theory

The energy expression is

$$E = \langle\Psi|H|\Psi\rangle = 2\sum_i h_{ii} + \sum_{ij}[2(ii|jj) - (ij|ij)] \tag{A.9}$$

with $\Psi = \mathcal{A}(\psi_1^2\psi_2^2\cdots\psi_n^2)$, where the superscript 2 indicates dual occupancy. Each orbital ψ_i is expressed in terms of the basis functions

$$\psi_i = \sum_{\alpha=1}^{m} c_{\alpha i}\eta_\alpha \tag{A.10}$$

The orbitals which make the energy stationary with respect to variations of the molecular orbital coefficients $c_{\alpha i}$, maintaining orbital orthonormality are then found.

If n of these orbitals ϕ_i have been found, there will be $(m-n)$ other orbitals ϕ_a (called unoccupied or virtual orbitals) which obey $\langle\phi_a|\phi_i\rangle = 0$ because there are m total basis functions. The condition that the energy is stationary with respect to the variation

$$\phi_k \longrightarrow \phi_k + \epsilon\phi_a \qquad (k = 1, 2, \ldots n; a = n+1, n+2, \ldots m) \tag{A.11}$$

is therefore found. Substituting A.11 in A.9 and setting everything with a coefficient of ϵ to zero gives the stationary condition. For the one electron part

$$\langle\phi_k + \epsilon\phi_a|h|\phi_k + \epsilon\phi_a\rangle = h_{kk} + \epsilon(h_{ak} + h_{ka}) + \epsilon^2 h_{aa} \tag{A.12}$$

the coefficient is therefore $2h_{ak}$ (using hermiticity). Similarly for the two electron part

$$\begin{aligned}
(k+\epsilon a\, k+\epsilon a|jj) &= (kk|jj) + \epsilon[(ka|jj) + (ak|jj)] + \cdots & \text{(A.13)}\\
(ii|k+\epsilon a\, k+\epsilon a) &= (ii|kk) + \epsilon[(ii|ka) + (ii|ak)] + \cdots & \text{(A.14)}\\
(k+\epsilon a\, j|k+\epsilon a\, j) &= (kj|kj) + \epsilon[(aj|kj) + (kj|aj)] + \cdots & \text{(A.15)}\\
(i\, k+\epsilon a|i\, k+\epsilon a) &= (ik|ik) + \epsilon[(ik|ia) + (ia|ik)] + \cdots & \text{(A.16)}
\end{aligned}$$

using the properties of two electron integrals and replacing $\sum_i$ by $\sum_j$ where appropriate, the stationary condition is obtained

$$4h_{ak} + \sum_j [8(ak|jj) - 4(aj|kj)] = 0 \tag{A.17}$$

or

$$h_{ak} + \sum_j [2(ak|jj) - (aj|kj)] = 0 \tag{A.18}$$

The Fock hamiltonian F is defined such that

$$\langle\phi_a|F|\phi_k\rangle = h_{ak} + \sum_j [2(ak|jj) - (aj|kj)] \tag{A.19}$$

which is more recognisable as a hamiltonian when written as

$$F(1) = h(1) + \sum_j 2 \int \frac{\phi_j^2(2)}{r_{12}} dr_2 - \sum_j \int \frac{\phi_j(2)\phi_j(1)}{r_{12}} dr_2 P_{12} \qquad \text{(A.20)}$$

where $P_{12}\phi_k(1) = \phi_k(2)$. The Fock hamiltonian is therefore an effective one-electron hamiltonian, including a kinetic term, nuclear attraction and an average potential term made up of a n electron coulomb part and an electron exchange part. Thus from A.18 and A.19, the condition that the energy is stationary with respect to variations of the molecular orbital coefficients is

$$F_{ak} \equiv (\phi_a|F|\phi_k) = 0 \qquad (k = 1, 2, \ldots n; a = n+1, n+2, \ldots m) \qquad \text{(A.21)}$$

The orbitals that satisfy this condition may be obtained by solving the canonical secular equations

$$\sum_\beta (\eta_\alpha|F - \epsilon_i|\eta_\beta) c_{\beta i} = 0 \qquad \text{(A.22)}$$

where ϵ_i is the i^{th} energy, for which it is known that the resulting orbitals obey

$$F_{pq} = \epsilon_p \delta_{pq} \qquad \text{(A.23)}$$

Thus the solutions of the secular equations obey the conditions of A.21. In practise, because F is a hamiltonian and ϵ_i are the corresponding energies of the orbitals ϕ_i, the n lowest eigensolutions of A.22 are identified as the occupied orbitals and the $(m-n)$ remaining solutions as unoccupied orbitals. F is defined in terms of its solution so an iterative procedure is required to solve the Self Consistent Field equations A.22:

i Select the geometry of the molecule and the basis set.

ii Evaluate the basis function integrals $h_{\alpha\beta}$,$(\alpha\beta|\gamma\delta)$ and $S_{\alpha\beta}$

iii Guess some coefficients $c_{\alpha\beta}$ for the occupied orbitals

iv Form the density matrix $D_{\alpha\beta} = \sum_i^n c_{\alpha i} c_{\beta i}$

v Construct the Fock matrix

$$(\eta_\alpha|F|\eta_\beta) = h_{\alpha\beta} + \sum_{\gamma\delta} (2(\alpha\beta|\gamma\delta) - (\alpha\gamma|\beta\delta)) \qquad \text{(A.24)}$$

vi Solve the secular equations A.22. Go to step (iv).

In step (iv), other than the first iteration, check that $\mathbf{D}$ has changed (to a sufficiently small tolerance) from the previous iteration; if it has, the equations have converged and the energy is then calculated using

$$E = 2\sum_{\alpha\beta} D_{\alpha\beta} h_{\alpha\beta} + \sum_{\alpha\beta\gamma\delta} D_{\alpha\beta} D_{\gamma\delta} (2(\alpha\beta|\gamma\delta) - (\alpha\gamma|\beta\delta)) \qquad \text{(A.25)}$$

A.2 Density Functional Theory

Since 1993 DFT has become the most often used approach of computational quantum chemistry for the study of ground state molecular properties. In DFT, the total energy is expressed in terms of the total electron density rather than the wave function. In this type of calculation, there is an approximate Hamiltonian and an approximate expression for the total electron density. DFT methods can be very accurate for comparatively little computational cost. The drawback is, that unlike *ab initio* methods, there is no systematic way to improve the methods by improving the form of the functional.

Physicists have been promoting the use of DFT since Slater's contribution in 1951 [212] which suggests the replacement of the exchange term in the Hartree-Fock method by the Dirac potential [213] which he argued contained both the exchange and correlation effects. This original form made the molecules significantly over bound. There are now no problems with matrix element evaluation and DFT codes which use local functionals are now less expensive to use than Hartree-Fock codes.

Traditional methods in electronic structure theory, in particular Hartree-Fock theory and its descendants, are based on the complicated many-electron wavefunction. The main objective of DFT is to replace the many-body electronic wavefunction with the electronic density as the basic quantity. Whereas the many-body wavefunction is dependent on $3N$ variables, three spatial variables for each of the N electrons, the density is only a function of three variables and is a simpler quantity to deal with both conceptually and practically.

If N is the number of elections then the density $\rho(\mathbf{r})$ is defined by

$$\rho(\mathbf{x_1}) = N \int \dots \int |\Psi|^2 ds_1 d\mathbf{x_2} \dots d\mathbf{x_N} \tag{A.26}$$

where $\Psi(\mathbf{x_1 x_2} \dots \mathbf{x_N})$ is the electronic wavefunction for the molecule. It is observed that

$$\int \rho(\mathbf{r}) d\mathbf{r} = N \tag{A.27}$$

The most common implementation of density functional theory is through the Kohn-Sham method [214]. The Kohn-Sham equations for the Kohn-Sham orbitals ϕ_i are

$$\left(-\frac{1}{2}\nabla^2 + v(\mathbf{r}) + \int \frac{\rho(\mathbf{r}')}{|\mathbf{r} - \mathbf{r}'|} d\mathbf{r}' + v_{xc}(\mathbf{r}) \right) \phi_i(\mathbf{r}) = \epsilon_i \phi_i(\mathbf{r}) \tag{A.28}$$

where v_{xc} is the exchange-correlation potential. If this can be exactly determined the exact density is accessible. However this remains unlikely and currently semi-empirical functionals are used instead. One of the most frequently used functionals is B3LYP [215, 216, 217].

B3LYP which is a hybrid functional in which the exchange energy, in this case from Becke's exchange functional, is combined with the exact energy from Hartree-Fock theory. Three parameters define the hybrid functional, specifying how much of the exact exchange is mixed in. The adjustable parameters in hybrid functionals are generally fitted to a training set of molecules. Unfortunately, although the results obtained with these functionals are usually sufficiently accurate for most applications, there is no systematic way of improving them (in contrast to some of the traditional wavefunction-based methods like configuration interaction or coupled cluster theory). Hence in the current DFT approach it is not possible to estimate the error of the calculations without comparing them to other methods or experiment.

Within the framework of Kohn-Sham DFT, the intractable many-body problem of interacting electrons in a static external potential is reduced to a tractable problem of non-interacting electrons moving in an effective potential. The effective potential includes the external potential and the effects of the Coulomb interactions between the electrons.

A.3 Semi-Empirical Methods

Semi-empirical quantum chemistry methods are based on the Hartree-Fock formalism but make many approximations and obtain some parameters from empirical data. They are very important in computational chemistry for treating large molecules where the full Hartree-Fock method without approximations is too expensive. The use of empirical parameters appears to allow some inclusion of electron correlation effects into the methods.

Within the framework of Hartree-Fock calculations, some pieces of information (such as two-electron integrals) are sometimes approximated or completely omitted. In order to correct for this loss, semi-empirical methods are parameterised. That is, their results are fitted by a set of parameters, normally in such a way as to produce results that best agree with experimental data, but sometimes to agree with *ab initio* results.

Semi-empirical calculations are much faster than *ab initio* methods but the results can be very wrong if the molecule being computed is not similar enough to the molecules in the database used to parameterise the method. Semi-empirical calculations have been most successful in the description of organic chemistry, where only a few elements are used extensively and molecules are of moderate size.

The AM1 (Austin Model 1), is a semi-empirical method for the quantum calculation of molecular electronic structure in computational chemistry. It is based on the Neglect of Differential Diatomic Overlap (NDDO) integral approximation [218]. Specifically, it is a generalization of the modified neglect of differential diatomic overlap (MNDO) approximation. AM1 was developed by Dewar and co-workers and published in 1985 [219].

AM1 is an attempt to improve the MNDO model by reducing the repulsion of atoms at small separation. The atomic core terms in the MNDO equations were modified through the addition of off-center attractive and repulsive Gaussian functions. The complexity of the parameterisation problem increased in AM1 as the number of parameters per atom increased from 7 in MNDO to 13-16 per atom in AM1.

The PM3 method (Parameterised Model 3) is based on the NDDO integral approximation. The PM3 method uses the same formalism and equations as the AM1 method. The only differences are that PM3 uses two Gaussian functions for the core repulsion function, instead of the variable number used by AM1 (which uses between one and four Gaussians per element) and that the numerical values of the parameters are different. Other differences lie in the methodology used during the parameterisation; whereas AM1 takes some of the parameter values from spectroscopical measurements, PM3 treats them as values which may be optimised.

The PM3 method was developed by Stewart and first published in 1989 [220, 221]. It is implemented in the MOPAC program, along with the related AM1, MNDO and MINDO methods. The original PM3 publication included parameters for the following elements: H, C, N, O, F, Al, Si, P, S, Cl, Br, and I. Many other elements, mostly metals, have subsequently been parameterised.

Appendix B

Regular Expressions in Java

The following is taken from the Java documentation on regular expressions as defined in the package java.util.regex [222]

Table B.1: Regular expression constructs as specified by Java

Summary of regular-expression constructs	
Construct	Matches
Characters	
χ	The character χ
\	The backslash character
\0*n*	The character with octal value 0*n* ($0 <= n <= 7$)
\0*nn*	The character with octal value 0*nn* ($0 <= n <= 7$)
\0*mnn*	The character with octal value 0*mnn* ($0 <= m <= 3$, $0 <= n <= 7$)
\x*hh*	The character with hexadecimal value 0x*hh*
\u*hhhh*	The character with hexadecimal value 0x*hhhh*
\t	The tab character (‘\u0009’)
\n	The newline (line feed) character (‘\u000A’)
\r	The carriage-return character (‘\u000D’)
\f	The form-feed character (‘\u000C’)
\a	The alert (bell) character (’\u0007’)
\e	The escape character (‘\u001B’)
\c$\chi$	The control character corresponding to χ
Character classes	
[abc]	a, b, or c (simple class)
[ˆabc]	Any character except a, b, or c (negation)
[a-zA-Z]	a through z or A through Z, inclusive (range)
[a-d[m-p]]	a through d, or m through p: [a-dm-p] (union)
[a-z&&[def]]	d, e, or f (intersection)
[a-z&&[ˆbc]]	a through z, except for b and c: [ad-z] (subtraction)
[a-z&&[ˆm-p]]	a through z, and not m through p: [a-lq-z](subtraction)

Continued on Next Page...

Table B.1 – Continued

Construct	Matches
Predefined character classes	
.	Any character (may or may not match line terminators)
\d	A digit: [0-9]
\D	A non-digit: [^0-9]
\s	A whitespace character: [\t\n\x0B\f\r]
\S	A non-whitespace character: [^\s]
\w	A word character: [a-zA-Z_0-9]
\W	A non-word character: [^\w]
POSIX character classes (US-ASCII only)	
\p{Lower}	A lower-case alphabetic character: [a-z]
\p{Upper}	An upper-case alphabetic character:[A-Z]
\p{ASCII}	All ASCII:[\x00-\x7F]
\p{Alpha}	An alphabetic character: [\p{Lower}\p{Upper}]
\p{Digit}	A decimal digit: [0-9]
\p{Alnum}	An alphanumeric character: [\p{Alpha}\p{Digit}]
\p{Punct}	Punctuation: One of !"#$%&'()*+,-./:;<=>?[\]^_`{|}~
\p{Graph}	A visible character: [\p{Alnum}\p{Punct}]
\p{Print}	A printable character: [\p{Graph}\x20]
\p{Blank}	A space or a tab: [\t]
\p{Cntrl}	A control character: [\x00-\x1F\x7F]
\p{XDigit}	A hexadecimal digit: [0-9a-fA-F]
\p{Space}	A whitespace character: [\t\n\x0B\f\r]
java.lang.Character classes (simple java character type)	
\p{javaLowerCase}	Equivalent to java.lang.Character.isLowerCase()
\p{javaUpperCase}	Equivalent to java.lang.Character.isUpperCase()
\p{javaWhitespace}	Equivalent to java.lang.Character.isWhitespace()
\p{javaMirrored}	Equivalent to java.lang.Character.isMirrored()
Classes for Unicode blocks and categories	
\p{InGreek}	A character in the Greek block (simple block)
\p{Lu}	An uppercase letter (simple category)
\p{Sc}	A currency symbol
\P{InGreek}	Any character except one in the Greek block (negation)
[\p{L}&&[^\p{Lu}]]	Any letter except an uppercase letter (subtraction)
Boundary matchers	
^	The beginning of a line
$	The end of a line
\b	A word boundary
\B	A non-word boundary
\A	The beginning of the input
\G	The end of the previous match
\Z	The end of the input but for the final terminator, if any
\z	The end of the input

Continued on Next Page...

Table B.1 – Continued

Construct	Matches
Greedy quantifiers	
$X?$	X, once or not at all
$X*$	X, zero or more times
$X+$	X, one or more times
$X\{n\}$	X, exactly n times
$X\{n,\}$	X, at least n times
$X\{n,m\}$	X, at least n but not more than m times
Reluctant quantifiers	
$X??$	X, once or not at all
$X*?$	X, zero or more times
$X+?$	X, one or more times
$X\{n\}?$	X, exactly n times
$X\{n,\}?$	X, at least n times
$X\{n,m\}?$	X, at least n but not more than m times
Possessive quantifiers	
$X?+$	X, once or not at all
$X*+$	X, zero or more times
$X++$	X, one or more times
$X\{n\}+$	X, exactly n times
$X\{n,\}+$	X, at least n times
$X\{n,m\}+$	X, at least n but not more than m times
Logical operators	
XY	X followed by Y
$X\|Y$	Either X or Y
(X)	X, as a capturing group
Back references	
$\backslash n$	Whatever the n^{th} capturing group matched
Quotation	
\	Nothing, but quotes the following character
\Q	Nothing, but quotes all characters until \E
\E	Nothing, but ends quoting started by \Q
Special constructs (non-capturing)	
(?:X)	X, as a non-capturing group
(?idmsux-idmsux)	Nothing, but turns match flags on - off
(?idmsux-idmsux:X)	X, as a non-capturing group with the given flags on - off

Backslashes, escapes, and quoting

The backslash character ('\') serves to introduce escaped constructs, as defined in the table above, as well as to quote characters that otherwise would be interpreted as unescaped constructs. Thus the expression \\ matches a single backslash and \{ matches a left brace.

It is an error to use a backslash prior to any alphabetic character that does not denote an escaped construct; these are reserved for future extensions to the regular-expression language. A backslash may be used prior to a non-alphabetic character regardless of whether that character is part of an unescaped construct.

Backslashes within string literals in Java source code are interpreted as required by the Java Language Specification as either Unicode escapes or other character escapes. It is therefore necessary to double backslashes in string literals that represent regular expressions to protect them from interpretation by the Java bytecode compiler. The string literal "\b", for example, matches a single backspace character when interpreted as a regular expression, while "\\b" matches a word boundary. The string literal "\(hello\)" is illegal and leads to a compile-time error; in order to match the string `(hello)` the string literal "\\(hello\\)" must be used.

Character Classes

Character classes may appear within other character classes, and may be composed by the union operator (implicit) and the intersection operator (&&). The union operator denotes a class that contains every character that is in at least one of its operand classes. The intersection operator denotes a class that contains every character that is in both of its operand classes.

The precedence of character-class operators is as follows, from highest to lowest:

1. Literal escape \`x`
2. Grouping [...]
3. Range `a-z`
4. Union [`a-e`][`i-u`]
5. Intersection [`a-z&&`[`aeiou`]]

Note that a different set of metacharacters are in effect inside a character class than outside a character class. For instance, the regular expression '.' loses its special meaning inside a character class, while the expression '-' becomes a range forming metacharacter.

Line terminators

A line terminator is a one- or two-character sequence that marks the end of a line of the input character sequence. The following are recognized as line terminators:

- A newline (line feed) character ('\n'),
- A carriage-return character followed immediately by a newline character ('\r\n'),
- A standalone carriage-return character ('\r'),
- A next-line character ('\u0085'),
- A line-separator character ('\u2028'), or
- A paragraph-separator character ('\u2029').

If UNIX_LINES mode is activated, then the only line terminators recognized are newline characters.

The regular expression '.' matches any character except a line terminator unless the DOTALL flag is specified.

By default, the regular expressions ˆ and $ ignore line terminators and only match at the beginning and the end, respectively, of the entire input sequence. If MULTILINE mode is activated then ˆ matches at the beginning of input and after any line terminator except at the end of input. When in MULTILINE mode $ matches just before a line terminator or the end of the input sequence.

Groups and capturing

Capturing groups are numbered by counting their opening parentheses from left to right. In the expression ((A)(B(C))), for example, there are four such groups:

1. ((A)(B(C)))
2. (A)

3. (B(C))
4. (C)

Group zero always stands for the entire expression.

Capturing groups are so named because, during a match, each subsequence of the input sequence that matches such a group is saved. The captured subsequence may be used later in the expression, via a back reference, and may also be retrieved from the matcher once the match operation is complete.

The captured input associated with a group is always the subsequence that the group most recently matched. If a group is evaluated a second time because of quantification then its previously-captured value, if any, will be retained if the second evaluation fails. Matching the string "aba" against the expression (a(b)?)+, for example, leaves group two set to "b". All captured input is discarded at the beginning of each match.

Groups beginning with (? are pure, non-capturing groups that do not capture text and do not count towards the group total.

Appendix C

Backus-Naur Form

The following definition for BNF is taken from The World of Programming Languages by Marcotty and Ledgard [122].

The meta-symbols of BNF are:

`::=` meaning *is defined as*

| meaning *or*

< > angle brackets used to surround category names

The angle brackets distinguish syntax rules names (also called non-terminal symbols) from terminal symbols which are written exactly as they are to be represented. A BNF rule defining a nonterminal has the form:

```
nonterminal ::= sequence_of_alternatives consisting
                of strings of terminals or nonterminals
                separated by the meta-symbol |
```

For example, the BNF production for a mini-language is:

```
<program> ::= program
                  <declaration_sequence>
              begin
                  <statements_sequence>
              end ;
```

This shows that a mini-language program consists of the keyword `program` followed by the declaration sequence, then the keyword `begin` and the statements sequence, finally the keyword `end` and a semicolon.

In fact, many authors have introduced some slight extensions of BNF for the ease of use:

- optional items are enclosed in meta symbols [and], for example

```
<if_statement> ::= if <boolean_expression> then
                       <statement_sequence>
                   [ else
                      <statement_sequence> ]
                   end if ;
```

- repetitive items (zero or more times) are enclosed in meta symbols **{** and **}**, for example

```
<identifier> ::= <letter> { <letter> | <digit> }
```

  this rule is equivalent to the recursive rule:

```
<identifier> ::= <letter> |
                 <identifier> [ <letter> | <digit> ]
```

- terminals of only one character are surrounded by quotes (‘’) to distinguish them from meta-symbols, for example:

```
<statement_sequence> ::=
       <statement> { ‘;’ <statement> }
```

- terminal and non-terminal symbols are distinguished by using bold faces for terminals and suppressing < and > around non-terminals. This improves greatly the readability. The example then becomes:

```
if_statement ::= if boolean_expression then
                   statement_sequence
                 [ else
                   statement_sequence ]
                 end if ‘;’
```

BNF’s syntax may be represented with a BNF like the following:

```
syntax     ::=  { rule }
rule       ::=  identifier  '::=' expression
expression ::=  term { '|' term }
term       ::= factor { factor }
factor     ::= identifier |
                 quoted_symbol |
                 '('  expression  ')' |
                 '['  expression  ']' |
                 '{'  expression  '}'
identifier ::=  letter { letter | digit }
quoted_symbol ::= ' ' ' { any_character } ' ' '
```

Appendix D

JFlex Lexical Rules

The syntax of the *lexical rules* section of a JFlex program is described by the following BNF grammar (terminal symbols are enclosed in ‘quotes’). This has been taken from the JFlex manual [132].

```
LexicalRules ::= Rule+
Rule         ::= [StateList] ['^'] RegExp [LookAhead] Action
               | [StateList] '<<EOF>>' Action
               | StateGroup
StateGroup   ::= StateList '{' Rule+ '}'
StateList    ::= '<' Identifier (',' Identifier)* '>'
LookAhead    ::= '$' | '/' RegExp
Action       ::= '{' JavaCode '}' | '|'

RegExp       ::= RegExp '|' RegExp
               | RegExp RegExp
               | '(' RegExp ')'
               | ('!'|'~') RegExp
               | RegExp ('*'|'+'|'?')
               | RegExp "{" Number [","  Number] "}"
               | '[' ['^'] (Character|Character'-'Character)* ']'
               | PredefinedClass
               | '{' Identifier '}'
               | '"' StringCharacter+ '"'
               | Character

PredefinedClass ::= '[:jletter:]'
                  | '[:jletterdigit:]'
                  | '[:letter:]'
                  | '[:digit:]'
                  | '[:uppercase:]'
                  | '[:lowercase:]'
                  | '.'
```

The grammar uses the following terminal symbols:

JavaCode a sequence of BlockStatements as described in the Java Language Specification.

Number a non negative decimal integer.

Identifier a letter [a-zA-Z] followed by a sequence of zero or more letters, digits or underscores [a-zA-Z0-9_]

Character an escape sequence or any unicode character that is not one of these meta characters: | ( ) { } [ ] < > \ . * + ? ^ $ / . " ~ !

StringCharacter an escape sequence or any unicode character that is not one of these meta characters: \ "

An escape sequence which consists of:

- \n \r \t \f \b

- a \x followed by two hexadecimal digits [a-fA-F0-9] (denoting a standard ASCII escape sequence)
- a \u followed by four hexadecimal digits [a-fA-F0-9] (denoting an unicode escape sequence)
- a backslash followed by a three digit octal number from 000 to 377 (denoting a standard ASCII escape sequence)
- a backslash followed by any other unicode character that stands for this character

Appendix E

GROMACS topology file

The GROMACS input parser was created by encoding all the allowable combinations shown in this appendix. The following is extracted from the GROMACS 3.1 manual [223] and is included for completeness. The topology file is built following the GROMACS specification for a molecular topology. All possible entries in the topology file are listed in Table E.1 and Table E.2. Also listed are all the units of the parameters, which interactions can be perturbed for free energy calculations, which bonded interactions are used by the GROMACS preprocessor (`grompp`) for generating exclusions and which bonded interactions can be converted to constraints by `grompp`. Description of the file layout:

- semicolon (;) and newline surround comments
- on a line ending with \ the newline character is ignored
- directives are surrounded by [and]
- the topology consists of three levels:
  - the parameter level (see Table E.1)
  - the molecule level, which should contain one or more molecule definitions (see TableE.2)
  - the system level: [`system`], [`molecules`]
- items should be separated by spaces or tabs, not commas
- atoms in molecules should be numbered consecutively starting at 1
- the file is parsed once only which implies that no forward references can be treated: items must be defined before they can be used
- exclusions can be generated from the bonds or overridden manually

- the bonded force types can be generated from the atom types or overridden per bond
- it is possible to apply multiple bonded interactions of the same type on the same atoms
- descriptive comment lines and empty lines are highly recommended
- starting with GROMACS version 3.1.3 all directives at the parameter level can be used multiple times and there are no restrictions on the order, except that an atom type needs to be defined before it can be used in other parameter definitions
- If parameters for a certain interaction are defined multiple times for the same combination of atom types the last definition is used; starting with GROMACS version 3.1.3 `grompp` generates a warning for parameter redefinitions with different values
- using one of the [`atoms`], [`bonds`], [`pairs`], [`angles`], *etc.* without having used [`moleculetype`] before is meaningless and generates a warning
- using [`molecules`] without having used [`system`] before is meaningless and generates a warning
- after [`system`] the only allowed directive is [`molecules`]
- using an unknown string in [ ] causes all the data until the next directive to be ignored, and generates a warning

Parameters

interaction type	directive	# at.	f. tp	parameters	F.E.
mandatory	`defaults`			non-bonded function type; combination rule$^{(\alpha)}$; generate pairs (no/yes); fudge LJ (); fudge QQ ();	
mandatory	atomtypes			atomtype;m(u);q(e);particle type; V$^{(\alpha)}$;W$^{(\alpha)}$	
	`bondtypes`	see table E.2, directive `bonds`			
	`constrainttypes`	see table E.2, directive `constraints`			
	`pairtypes`	see table E.2, directive `pairs`			
	`angletypes`	see table E.2, directive `angles`			
proper dih.	`dihedraltypes`	2/4$^{(b)}$	1	ϕ_s(deg);k_ϕ(kJ mol^{-1});multiplicity	ϕ,k
improper dih.	`dihedraltypes`	2/4$^{(c)}$	2	ζ_0(deg);k_ζ(kJ mol^{-1} rad^{-2})	all
RB dihedral	`dihedraltypes`	2/4$^{(b)}$	3	C_0, C_1, C_2, C_3, C_4, C_5 (kJ mol^{-1})	all
LJ	`nonbond_params`	2	1	V$^{(\alpha)}$;W$^{(\alpha)}$	
Buckingham	`nonbond_params`	2	2	a (kJ mol^{-1}); b (nm^{-1}); c_6 (kJ mol^{-1} nm^6)	

Molecule definition(s)

interaction type	directive	# at.	f. tp	parameters	F.E.
mandatory	`moleculetype`			moleculename exclude neighbours # bonds away for non-bonded interactions	
mandatory	`atoms`	1		atomtype; residue number; residue name; atom name; charge group number; q(e); m(u)	type q,m

intramolecular interaction definitions as described in table E.2

System

mandatory	`system`	system name
mandatory	`molecule`	molecule name; number of molecules

'# at.' is the number of atom types
'f. tp' is function type
'F.E' indicates which parameters can be interpolated during free energy calculations
$^{(a)}$ the combination rule determines the type of LJ parameters
$^{(b)}$ the inner two or all four atoms in the dihedral
$^{(c)}$ the outer two or all four atoms in the dihedral
For free energy calculations, the parameters for topology 'B' ($\lambda = 1$) should be added on the same line, after the normal parameters, in the same order as the normal parameters.

Table E.1: The topology file

Intramolecular interaction definitions

interaction type	directive	# at.	f. tp	parameters	F.E.
bond	`bonds`$^{(b,c)}$	2	1	b_0 (nm); k_b (kJ mol^{-1} nm^{-2})	all
G96 bond	`bonds`$^{(b,c)}$	2	1	b_0 (nm); k_b (kJ mol^{-1} nm^{-4})	all
morse	`bonds`$^{(b,c)}$	2	3	b_0 (nm); D (kJ mol^{-1});β (nm^{-1})	
cubic bond	`bonds`$^{(b,c)}$	2	4	b_0 (nm); $C_{i=2,3}$ (kJ mol^{-1} nm^{-i})	
connection	`bonds`$^{(b)}$	2	5		
harmonic pot.	`bonds`	2	6	b_0 (nm); k_b (kJ mol^{-1} nm^{-2})	all
FENE bond	`bonds`	2	7	b_m (nm); k_b (kJ mol^{-1} nm^{-2})	
LJ/Coul. 1-4	`pairs`	2	1	$V^{(a)}$; $W^{(a)}$	all
LJ/C. 1-4 A	`pairs`	2	2	$V^{(a)}$; $W^{(a)}$	
LJ/C. pair A	`pairs`	2	3		
angle	`angles`$^{(c)}$	3	1	θ_0 (deg); k_θ (kJ mol^{-1} rad^{-2})	all
G96 angle	`angles`$^{(c)}$	3	2	θ_0 (deg); k_θ (kJ mol^{-1})	all
quartic angle	`angles`$^{(c)}$	3	6	θ_0 (deg); $C_{i=0,1,2,3,4}$ (kJ mol^{-1} rad^{-i})	
proper dih.	`dihedrals`	4	1	ϕ_s (deg); k_ϕ (kJ mol^{-1}); multiplicity	all
improper dih.	`dihedrals`	4	2	ζ_0 (deg); k_ζ (kJ mol^{-1} rad^{-2})	all
RB dihedral	`dihedrals`	4	3	C_0,C_1,C_2,C_3,C_4,C_5 (kJ mol^{-1})	all
constraint	`constraints`$^{(b)}$	2	1	b_0 (nm)	all
constr. n.c.	`constraints`	2	2	b_0 (nm)	all
settle	`settles`	3	1	$d_{OH}\,d_{HH}$, (nm)	
vsite2	`virtual_sites2`	3	1	a ()	
vsite3	`virtual_sites3`	4	1	a, b ()	
vsite3fd	`virtual_sites3`	4	2	a (); d (nm)	
vsite3fad	`virtual_sites3`	4	3	θ (deg); d (nm)	
vsite3out	`virtual_sites3`	4	4	a, b (); c (nm^{-1})	
vsite4fd	`virtual_sites4`	5	1	a, b (); d (nm)	
position res.	`position_restraints`	1	1	k_x, k_y, k_z, (kJ mol^{-1} nm^{-2})	all
distance res.	`distance_restraints`	2	1	type; label; low, up$_1$, up$_2$ (nm); weight ()	
orient. res.	`orientation_restraints`	2	1	exp.; label; α; c (U nm$^\alpha$); obs. (U); weight (U^{-1})	
angle res.	`angle_restraints`$^{(c)}$	4	1	θ_0 (deg); k_c (kJ mol^{-1}); multiplicity	θ, k
angle res. z	`angle_restraints_z`$^{(c)}$	2	1	θ_0 (deg); k_c (kJ mol^{-1}); multiplicity	θ, k
exclusions	`exclusions`$^{(c)}$	1		one or more atom indicies	

'# at.' is the number of atom types
'f. tp' is function type
'F.E' indicates which parameters can be interpolated during free energy calculations
$^{(a)}$ the combination rule determines the type of LJ parameters
$^{(b)}$ used by `grompp` for generating exclusions
$^{(c)}$ can be converted to constraints by `grompp`
For free energy calculations, the parameters for topology 'B' ($\lambda = 1$) should be added on the same line, after the normal parameters, in the same order as the normal parameters.

Table E.2: Intramolecular actions definitions

Appendix F

Solvents and counter ions

There follows a list of all the structures deemed to be solvent/ion/guest molecules.

Inorganic molecules and counter ions

- SiF_6
- CO_3^{2-}
- CO_3H^-
- CO_3H_2
- NO_3
- HNO_3^+
- $SO4^{2-}$
- $HSO4^-$
- H_2SO4
- PF_6^-
- PO_4^{3-}
- HPO_4^{2-}
- $H_2PO_4^-$
- PO_3^{3-}
- HPO_3^{2-}

- $H2PO_3^-$
- $H3PO_3$
- ClO_3^-
- ClO_4^-
- $HClO_3$
- $HClO_4$
- BrO_3^-
- BrO_4^-
- $HBrO_3$
- $HBrO_4$
- IO_3^-
- IO_4^-
- HIO_3
- HIO_4
- BF_4^-

Small solvent molecules

All acids and alcohols are included in both their protonated and deprotonated forms.

- trichloromethane
- dicyanoamine
- oxalic acid (ethandioic acid)
- acetic acid
- fluorinated acetic acid
- chlorinated acetic acid
- brominated acetic acid
- sulfonic acid

- fluorinated sulfonic acid
- chlorinated sulfonic acid
- brominated sulfonic acid
- trifluoroethanol
- trichloroethanol
- tribromoethanol
- propanol
- acetone
- dimethylsulfoxide
- ether
- furan
- tetrahydrofuran
- *N*,*N*-dimethylformamide
- trimethylammonia
- trimethylammonium
- dimethyl sulphate
- amino ethanioc acid
- benzene
- toluene

Included for completeness

These molecules are too small (fewer than four heavy atoms) to be suitable for calculation, but are included for completeness.

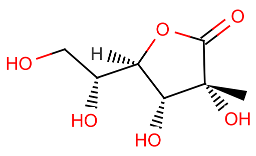
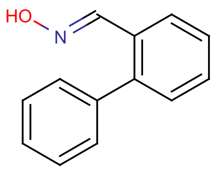
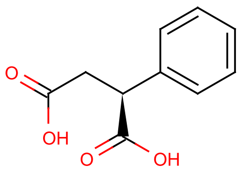
- F^-
- Cl^-
- Br^-
- I^-

- $NH4^+$
- dichloromethane
- hydrogencyanide
- CN^-
- H_2O

Appendix G

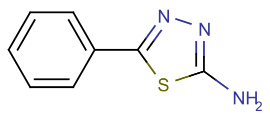
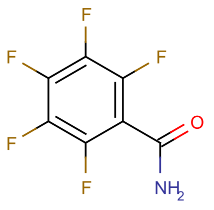
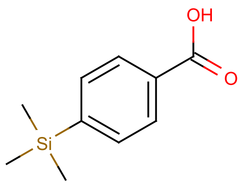
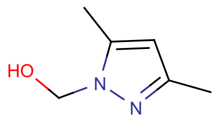
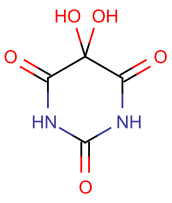
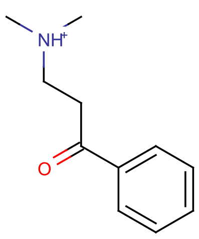
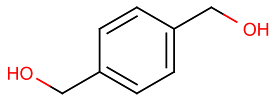
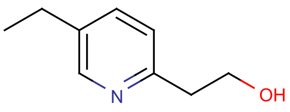
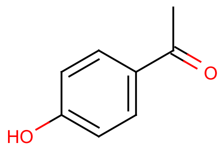
Molecules

Table G.1: The 112 unique connection tables of the molecules that fulfil the final protocol. The number of occurrences of each of the connection tables in the final dataset are indicated.

Molecules that pass the final protocol		
		
1	2	2

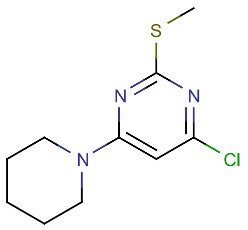
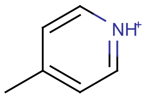
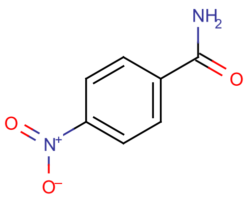
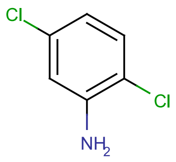
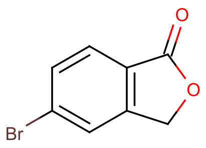
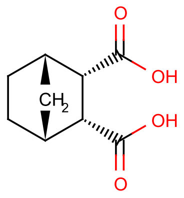
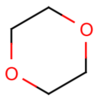
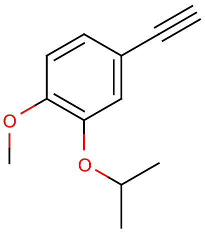
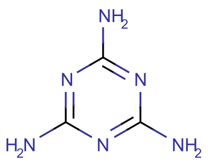
Continued on Next Page...

Table G.1 – Continued

		
1	1	1
		
1	1	1
		
1	1	1

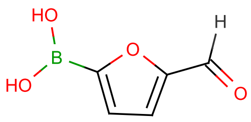
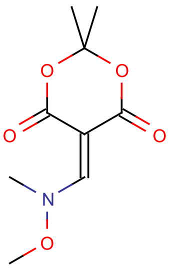
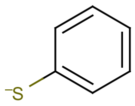
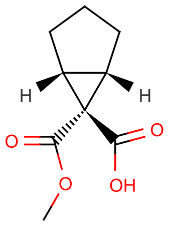
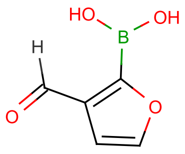
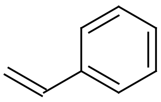
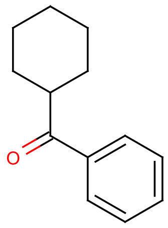
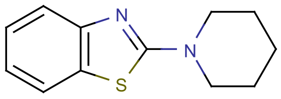
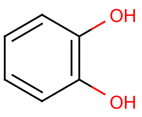
Continued on Next Page…

Table G.1 – Continued

		
1	1	1
		
1	1	2
		
2	1	1

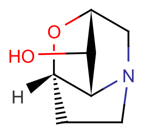
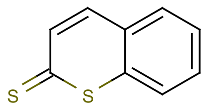
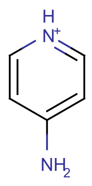
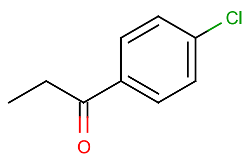
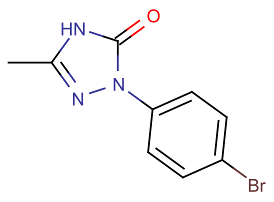
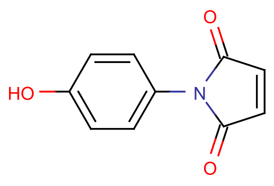
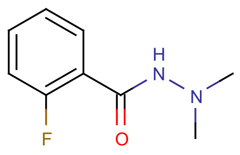
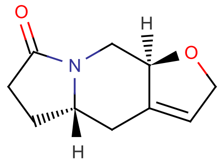
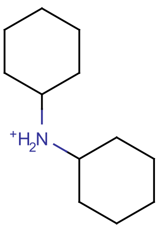
Continued on Next Page...

Table G.1 – Continued

		
1	1	1
		
1	2	2
		
1	1	1

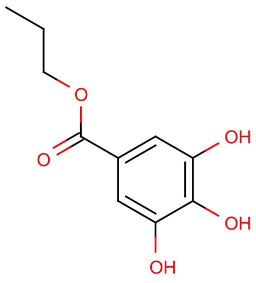
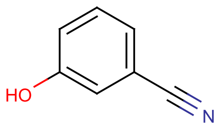
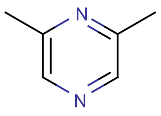
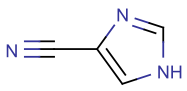
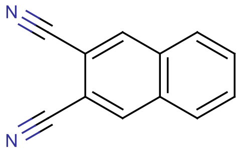
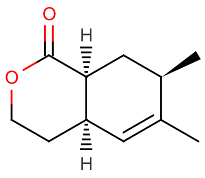
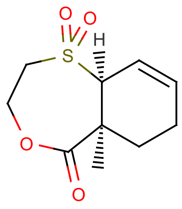
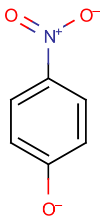
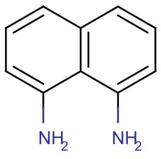
Continued on Next Page…

Table G.1 – Continued

 1	 1	 1
 1	 1	 1
 1	 1	 1

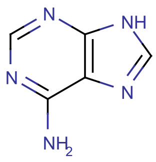
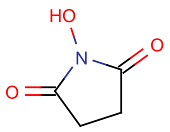
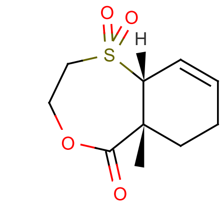
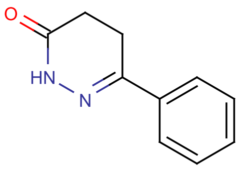
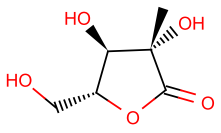
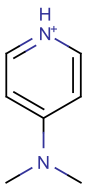
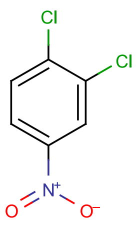
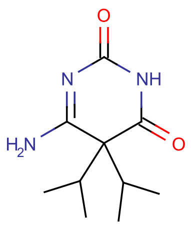
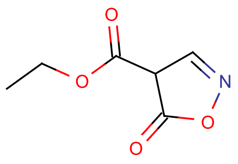
Continued on Next Page…

Table G.1 – Continued

 2	 1	 2
 1	 1	 1
 1	 1	 1

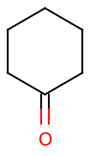
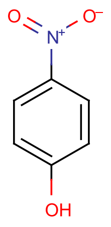
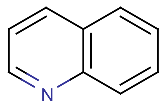
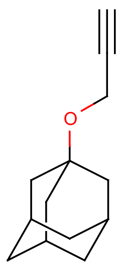
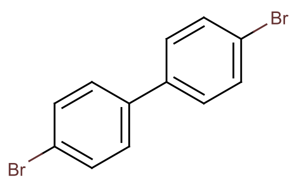
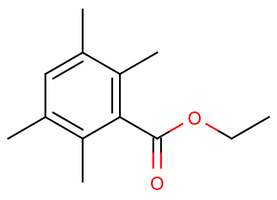
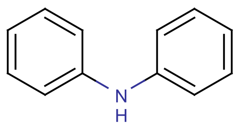
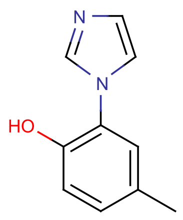
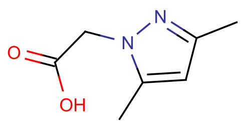
Continued on Next Page...

Table G.1 – Continued

		
1	1	1
		
1	1	3
		
2	2	1

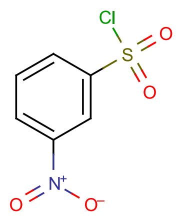
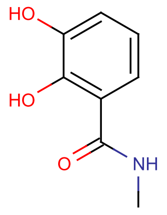
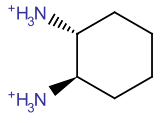
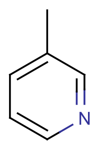
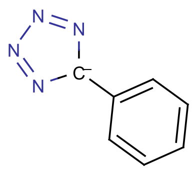
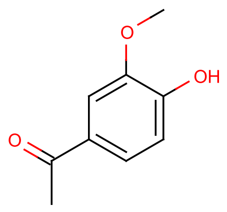
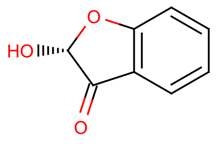
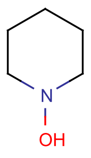
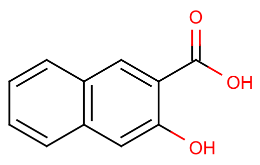
Continued on Next Page…

Table G.1 – Continued

		
1	1	2
		
1	1	1
		
1	1	1

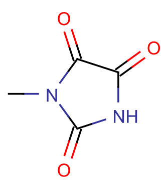
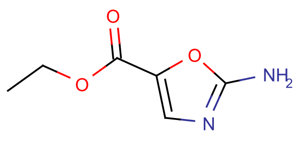
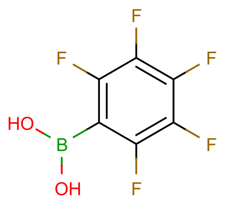
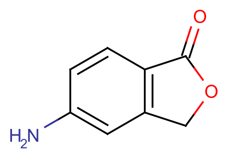
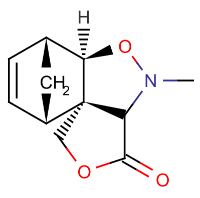
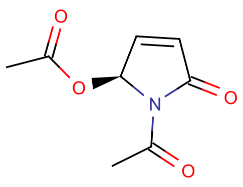
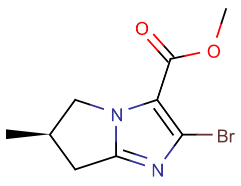
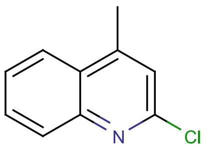
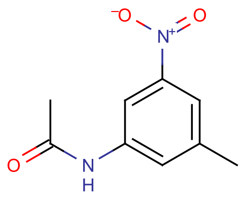
Continued on Next Page…

Table G.1 – Continued

		
1	1	1
		
1	1	1
		
1	1	1

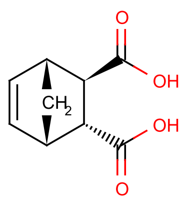
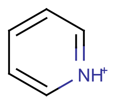
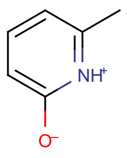
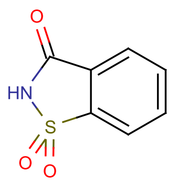
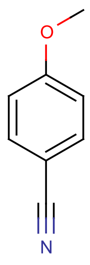
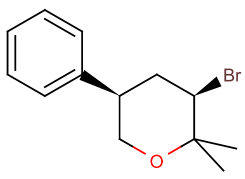
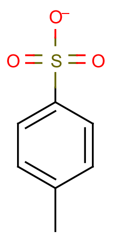
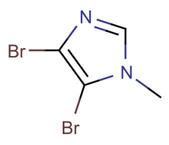
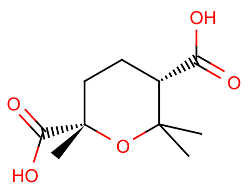
Continued on Next Page…

Table G.1 – Continued

		
1	1	1
		
1	1	1
		
1	1	1

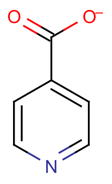
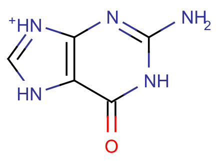
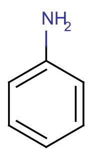
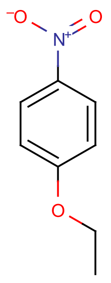
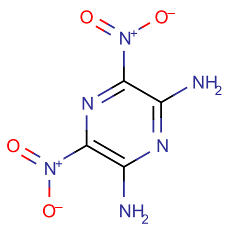
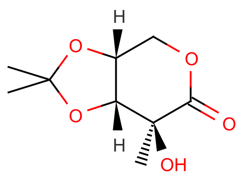
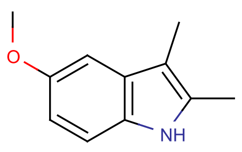
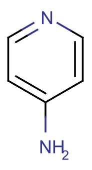
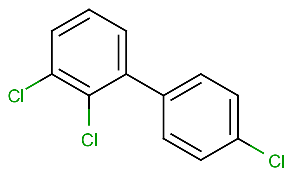
Continued on Next Page…

Table G.1 – Continued

 1	 1	 2
 1	 1	 1
 2	 1	 1

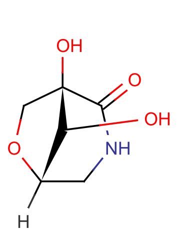
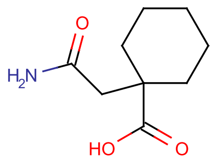
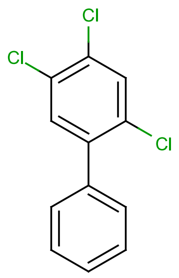
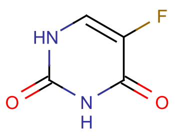
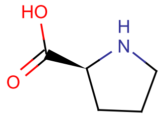
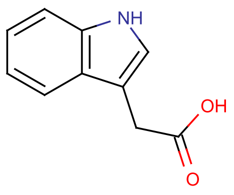
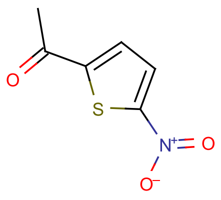
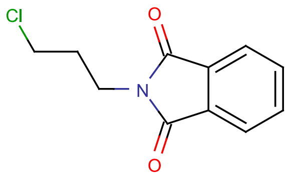
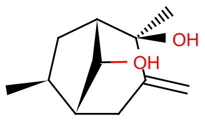
Continued on Next Page…

Table G.1 – Continued

		
1	1	1
		
1	1	1
		
1	1	1

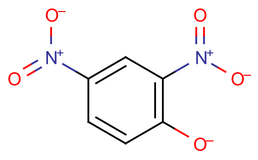
Continued on Next Page…

Table G.1 – Continued

		
1	1	1
		
3	1	1
		
1	1	1

Continued on Next Page...

Table G.1 – Continued

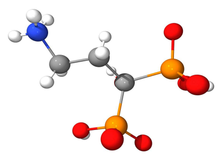
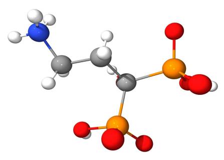
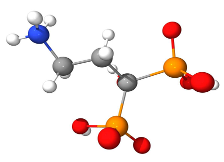
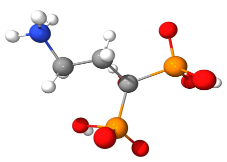
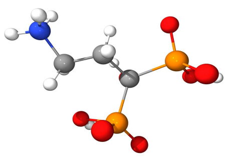
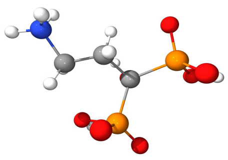
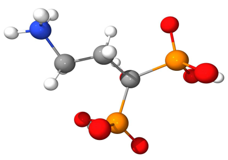
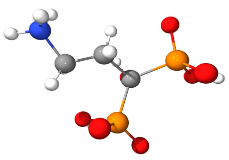
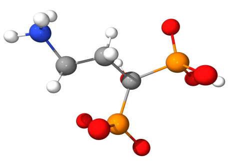
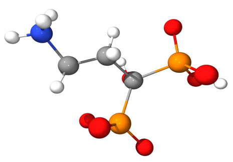
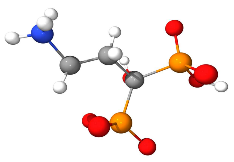
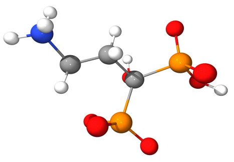
	
1	

Appendix H

Molecule Optimisations

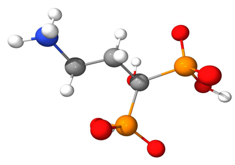
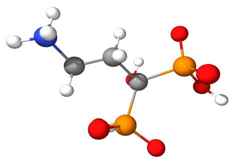
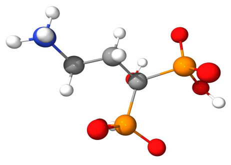
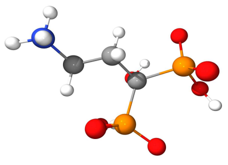
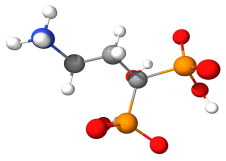
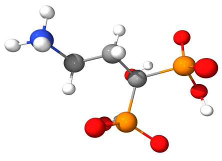
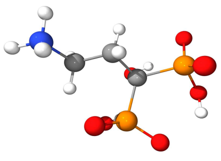
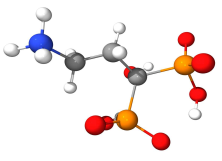
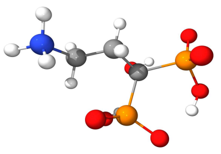
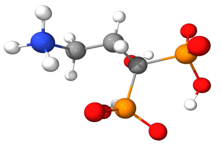
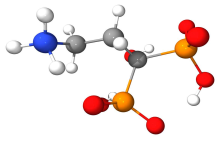
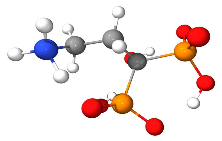
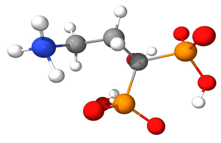
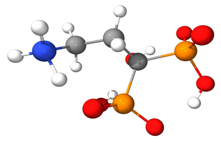
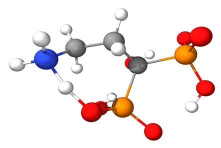
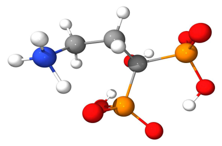
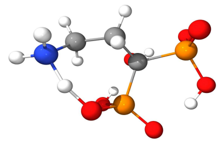
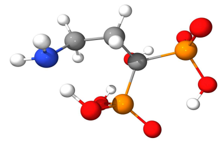
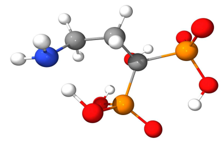
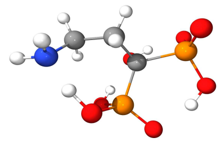
Table H.1: The reported geometries adopted by bv6006molecule3 during the geometry optimisation calculation. All steps of the optimisation are shown.

Geometry optimisation of bv6006molecule3

			
step 0	step 1	step 2	step 3
			
step 4	step 5	step 6	step 7
			
step 8	step 9	step 10	step 11

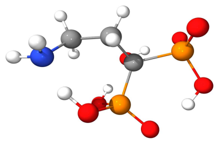
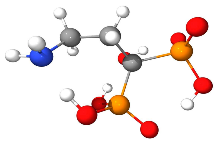
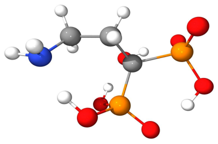
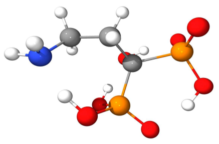
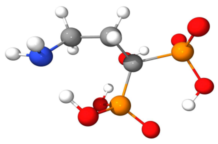
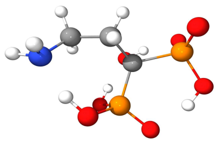
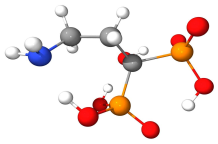
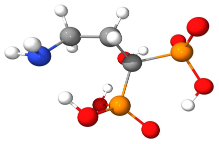
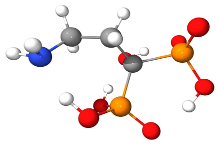
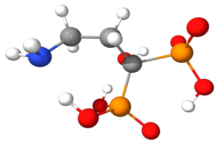
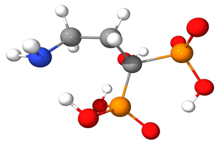
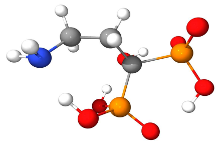
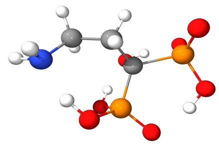
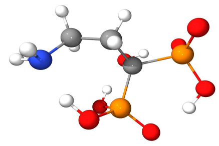
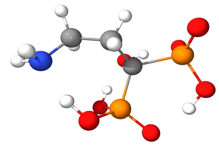
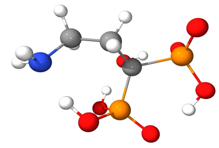
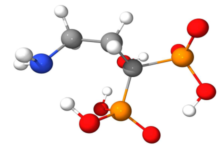
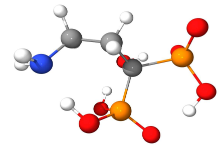
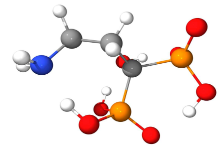
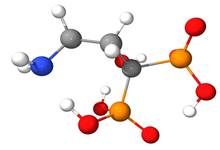
Continued on Next Page...

Table H.1 – Continued

 step 12	 step 13	 step 14	 step 15
 step 16	 step 17	 step 18	 step 19
 step 20	 step 21	 step 22	 step 23
 step 24	 step 25	 step 26	 step 27
 step 28	 step 29	 step 30	 step 31

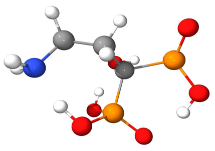
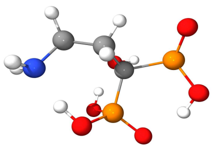
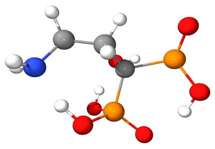
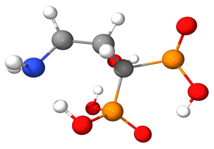
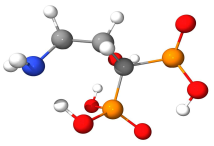
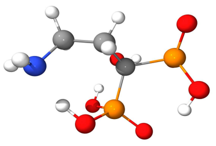
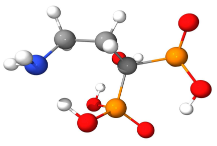
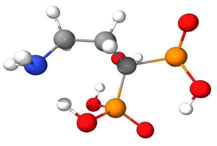
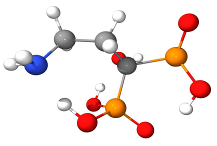
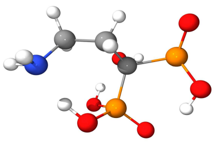
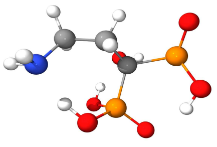
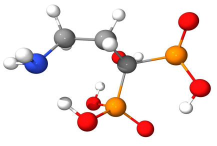
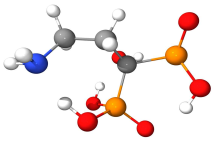
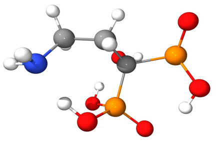
Continued on Next Page. . .

Table H.1 – Continued

 step 32	 step 33	 step 34	 step 35
 step 36	 step 37	 step 38	 step 39
 step 40	 step 41	 step 42	 step 43
 step 44	 step 45	 step 46	 step 47
 step 48	 step 49	 step 50	 step 51

Continued on Next Page…

Table H.1 – Continued

step 52	step 53	 step 54	 step 55
 step 56	 step 57	step 58	step 59
step 60	step 61	 step 62	 step 63
 step 64	 step 65	 step 66	 step 67
 step 68	 step 69	 step 70	 step 71

Continued on Next Page. . .

Table H.1 – Continued

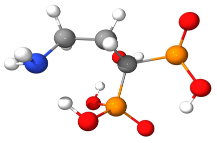
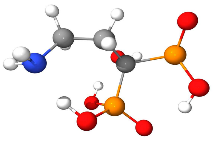
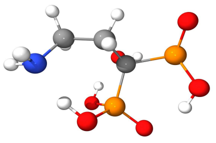
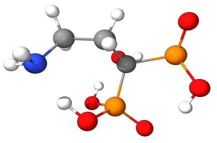
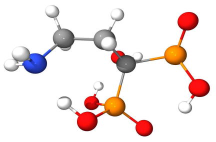
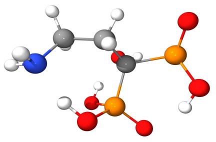
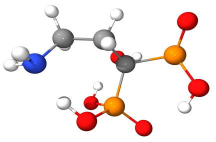
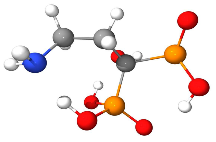
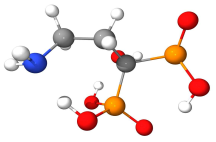
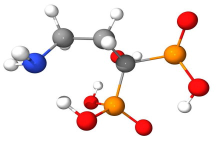
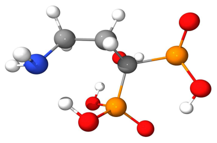
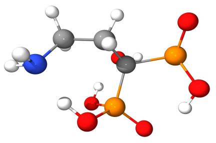
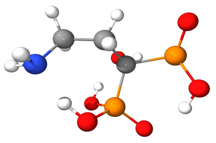
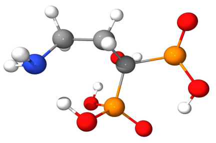
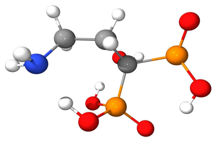
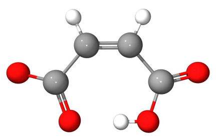
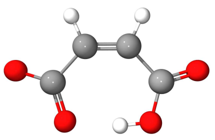
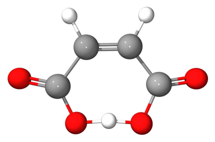
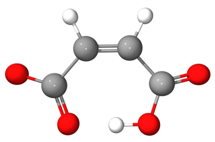
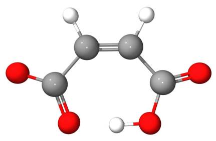
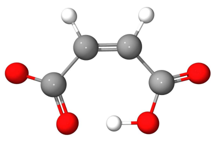
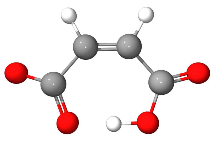
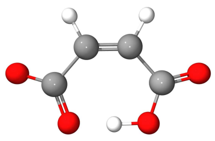
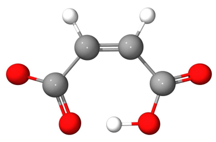
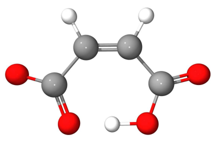
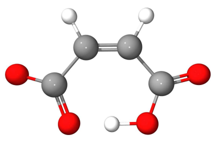
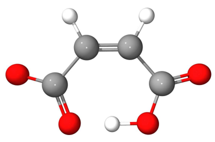
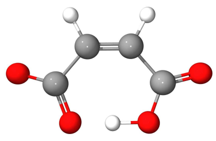
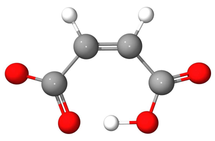
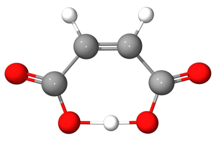
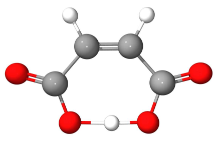
 step 72	 step 73	 step 74	 step 75
step 76	 step 77	 step 78	 step 79
step 80	 step 81	 step 82	 step 83
 step 84	 step 85	 step 86	 step 87
 step 88			

Table H.2: The reported geometries adopted by ci6067molecule2 during the geometry optimisation calculation. The first 20 (of 27) geometries are shown, no further protean behaviour was observed after this point.

Geometry optimisation of ci6067molecule2

			
step 0	step 1	step 2	step 3
			
step 4	step 5	step 6	step 7
			
step 8	step 9	step 10	step 11
			
step 12	step 13	step 14	step 15

Continued on Next Page…

Table H.2 – Continued

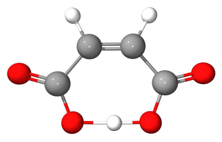
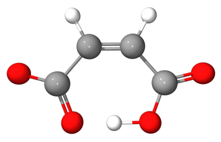
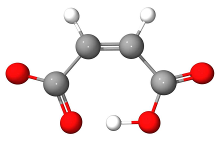
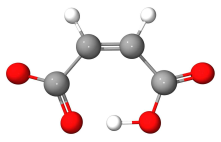
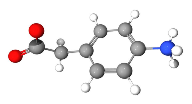
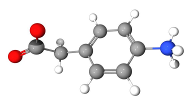
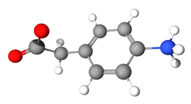
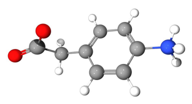
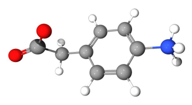
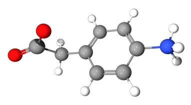
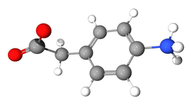
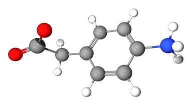
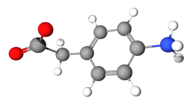
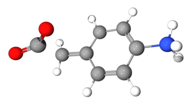
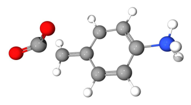
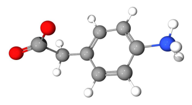
			
step 16	step 17	step 18	step 19

Table H.3: The reported geometries of rz6070molecule1 during the geometry optimisation calculation. The first 12 (of 52) geometries are shown, no further protean behaviour was observed after this point.

Geometry optimisation of rz6070molecule1			
			
step 0	step 1	step 2	step 3
			
step 4	step 5	step 6	step 7
			
step 8	step 9	step 10	step 11

Appendix I

Published Work

The following papers and communications have been published as a result of work contained in this thesis.

P. Murray-Rust, R. C. Glen, H. S. Rzepa, J. J. P. Stewart, J. A. Townsend, E. L. Willighagen, Y. Zhang, A semantic GRID for molecular science, *Proceedings of UK e-Science All Hands Conference* 2003

Y. Zhang, P. Murrary-Rust, M. T. Dove, R. C. Glen, H. S. Rzepa, J. A. Townsend, S. Tyrrell, J. Wakelin, E. L. Willighagen, JUMBO – An XML Infrastructure for eScience, *Proceedings of UK e-Science All Hands Conference* 2004

S. E. Adams, J. M. Goodman, R. J. Kidd, A. D. McNaught, P. Murray-Rust, F. R. Norton, J. A. Townsend, C. A. Waudby, Experimental data checker: better information for organic chemists, *Org. Biomol. Chem.*, **2004**, *2*, 3067–3070

J. A. Townsend, S. E. Adams, C. A. Waudby, V. K. de Souza, J. M. Goodman, P. Murray-Rust, Chemical documents: machine understanding and automated information extraction, *Org. Biomol. Chem.*, **2004**, *2*, 3294–3300

J. A. Townsend, P. Murray-Rust, Capturing chemistry in XML, *Abstr. Am.*

Chem. Soc. 2004

Y. Zhang, R. C. Glen, P. Murray-Rust, H. S. Rzepa, J. A. Townsend, Semantic grid computing — The WorldWideMolecularMatrix, *Abstr. Am. Chem. Soc.* 2004

J. A. Townsend, P. Murray-Rust, S. M. Tyrrell, Y. Zhang, Computational chemistry robots, *Abstr. Am. Chem. Soc.* 2005

P. Murray-Rust, H. S. Rzepa, J. A. Townsend, D. Wilson, Computational chemistry in XML, *Abstr. Am. Chem. Soc.* 2006

P. T. Corbett, P. Murray-Rust, N. E. Day, J .A. Townsend, H. S. Rzepa, Chemistry publications in CML, *Abstr. Am. Chem. Soc.* 2006

Bibliography

[1] T. Hey, A. Trefethen, The Data Deluge: An e-Science Perspective, *Wiley*, **2003**, 809–824

[2] M. Lesk, Practical digital libraries: Books, bytes, and bucks, *Morgan Kaufmann* 1997

[3] M. Atkinson, Grid Infrastructure meets Biological Research Challenges, **2002**, http://www.nesc.ac.uk/presentations/

[4] F. Berman, Viewpoint: From TeraGrid to knowledge grid, *Comm. ACM*, **2001**, *44*, 27–28

[5] M.R. Helal, Y.A. Yousef A.T. Afaneh, Ab Initio Calculations of the Stabilization Energies of the Conformational and the Structural Isomers of C_3H_7X where X = F, Cl, and Br, *J. Comp. Chem.*, **2003**, *23*, 966–976

[6] J.E. Davies, oral presentation, Unpublished Structures, *CCG Autumn Meeting*, **2003**

[7] F. Allen, oral presentation, The Future of Crystallographic 'Publication', *CCG Autumn Meeting*, **2003**

[8] http://www.admin.cam.ac.uk/offices/gradstud/current/submitting/phd/cdrom.html

[9] http://hdl.handle.net/1842/433

[10] http://wwmm.ch.cam.ac.uk/blogs/murrayrust/?p=362

[11] P. Murray-Rust, Data-driven Science — A Scientit's View, *NSF/JISC Repositories Workshop*, **2007**

[12] http://www.cas.org/newsevents/releases/milliondocs1206.html

[13] http://www.cas.org/ASSETS/836E3804111B49BFA28B95BD1B40CD0F/casstats.pdf

[14] H. Shojaei, Z. Li-Bohmer, P. vonZezschwitz, Iromycins: A New Family of Pyridone Metabolites from Streptomyces sp. II. Convergent Total Synthesis, *J. Org. Chem.*, **2007**, *72*, 5091–5097

[15] http://en.wikipedia.org/wiki/Metadata

[16] http://dublincore.org

[17] http://www.iupac.org/inchi

[18] J. J. P. Stewart, On the use of Semiempirical Methods for Detecting Anomalies in Reported Heats of Formation of Organic Compounds, *J. Phys. Chem. Ref. Data*, **2004**, *33*, 713–724

[19] B. Schlegel, A. Härtl, H. M. Dahse, F. A. Gollmick, U. Gräfe, H. Dörfelt H, B. Kappes, Hexacyclinol, a new antiproliferative metabolite of Panus rudis HKI 0254, *J. Antibiot.*, **2002**, *55*, 814–817

[20] J. J. La Clair, Total Syntheses of Hexacyclinol, 5-epi-Hexacyclinol, and Desoxohexacyclinol Unveil an Antimalarial Prodrug Motif, *Angew. Chem. Int. Ed.*, **2006**, *45*, 2769–2773

[21] S. D. Rychnovsky, Predicting NMR Spectra by Computational Methods: Structural Revision of Hexacyclinol, *Org. Lett.*, **2006**, *8*, 2895–2898

[22] http://www.nesc.ac.uk/nesc/mission.html

[23] T. Berners-Lee, J. Hendler, O. Lassila, The Semantic Web, *Sci. Am.*, **2001**, *284*, 34–43

[24] G. V. Gkoutous, P. Murray-Rust, H. S. Rzepa, M Wright, Chemical Markup, XML, and the World-Wide Web. 3. Toward a Signed Semantic Chemical Web of Trust, *J. Chem. Inf. Comput. Sci.*, **2001**, *41*, 1124–1130

[25] P. Murray-Rust, H. S. Rzepa, M. J. Williamson, E. L. Willighagen, Chemical Markup, XML, and the World-Wide Web. 5. Applications of Chemical Metadata in RSS Aggregators, *J. Chem. Inf. Comput. Sci.*, **2004**, *44*, 462–469

[26] R. D. King, M. Young, A. J. Clare, K. E. Whelan, J. Rowland, The Robot Scientist Project, *Springer Berlin* **2005**

[27] T. Helgaker, P. Jørgensen, J. Olsen, Molecular Electronic-Structure Theory, *Wiley* 2004

[28] R. R. Gotwals, S. C. Sendlinger, A Chemistry Educator's Guide to Molecular Modeling, **2007**, http://chemistry.ncssm.edu/book/

[29] http://www.w3.org/TR/REC-xml

[30] J. H. Coombs, A. H. Renear, S. J. DeRose, Markup Systems and the Future of Scholarly Text Processing, *Comm. ACM*, **1987**, *30*, 933–947

[31] http://www.w3.org/TR/html401

[32] http://www.w3.org

[33] http://www.w3.org/TR/xhtml1

[34] http://www.w3.org/TR/REC-xml/#dt-doctype

[35] http://www.w3.org/XML/Schema

[36] http://www.w3.org/TR/REC-xml-names

[37] P Murray-Rust, H. S. Rzepa, STMML. A markup language for scientific, technical and medical publishing, *Data Sci.*, **2002**, *1*, 128–192

[38] NISO Standard Z39.85-2001, http://www.niso.org

[39] ISO Standard 15836-2003, http://www.iso.org

[40] http://www.w3.org/TR/xslt

[41] P. Murray-Rust, H. S. Rzepa, Handbook of Chemoinformatics, *Wiley-VCH*, 2003

[42] http://www.xml-cml.org/information/position.html

[43] P. Murray-Rust, H. S. Rzepa, Chemical markup, XML, and the World-wide Web. 1. Basic principles, *J. Chem. Inf. Comput. Sci.*, **1999**, *39*, 928–942

[44] A. Dalby, J. G. Nourse, W. Douglas Hounshell, A. K. I. Gushurst, D. L. Grier, A. Leland, J. Laufer, Description of several chemical structure file formats used by computer programs developed at Molecular Design Limited, *J. Chem. Inf. Comput. Sci.*, **1992**, *32*, 244–255

[45] P. Murray-Rust, H. S. Rzepa, Chemical markup, XML, and the World-wide Web. 4. CML Schema, *J. Chem. Inf. Comput. Sci.*, **2003**, *43*, 757–772

[46] G. W. Kramer, ANIML: Analytical information markup language for spectroscopy and chromatography data, *Abstr. Am. Chem. Soc.* 2003

[47] http://www.gaml.org

[48] A. D. T. Nguyen, A. Arslan, J. Travis, M. Smith, R. Schäfer, G. W.Kramer, Molecular spectrometry data interchange applications for NIST's SpectroML, *J. Assoc. Lab. Auto.*, **2004**, *9*, 346–354

[49] J. Wakelin, P. Murray-Rust, S. M. Tyrrell, Y. Zhang, H. S. Rzepa, A. Garcia, *Mol. Simulations*, **2005**, *31*, 315–322

[50] http://www.w3.org/TR/REC-DOM-Level-1

[51] http://www.w3.org/TR/DOM-Level-2-Core

[52] P Murray-Rust, H. S. Rzepa, Chemical markup, XML, and the World-wide Web. 2. Information Objects and the CMLDOM, *J. Chem. Inf. Comput. Sci.*, **2001**, *41*, 1113–1123

[53] D. E. Knuth, The Art of Computer Programming, *Addison-Wesley*, 1997

[54] http://www.opensource.org/docs/osd

[55] http://sourceforge.net/projects/cml

[56] http://www.w3.org/Graphics/SVG

[57] http://www.sun.com

[58] http://www.adobe.com

[59] http://www.apple.com

[60] http://www.ibm.com

[61] http://www.kodak.com

[62] http://www.w3.org/TR/xlink

[63] http://www.w3.org/TR/xmlbase

[64] http://www.w3.org/TR/xml-stylesheet

[65] http://www.w3.org/TR/REC-smil

[66] http://www.w3.org/TR/2001/REC-smil-animation-20010904

[67] http://www.r-project.org

[68] http://office.microsoft.com/excel

[69] http://www.uszla.me.uk/software/pelote.html

[70] J. Bishop, Java Gently: Programming Principles Explained, *Addison-Wesley* 1998

[71] http://java.sun.com

[72] http://jmol.sourceforge.net

[73] http://www.povray.org

[74] http://www.winedt.com

[75] http://office.microsoft.com/word

[76] C. Creighton, S. Hanash, Mining gene expression databases for association rules, *Bioinformatics*, **2003**, *19*, 79–86

[77] M. Andrade, A. Valencia, Automatic extraction of keywords from scientific text: Application to the knowledge domain of protein families, *Bioinformatics*, **1998**, *14*, 600–607

[78] J. M. Temkin, M. R. Gilder, Extraction of protein interaction information from unstructured text using a context-free grammar, *Bioinformatics*, **2003**, *19*, 2046–2053

[79] P. Murray-Rust, H. S. Rzepa, The Next Big Thing: From Hypermedia to Datuments, *J. Digital Information*, **2004**, *5*, 248

[80] L. R. Garson, Communicating original research in chemistry and related sciences, *Acc. Chem. Res.*, **2004**, *37*, 141–148

[81] F. Damerau, A technique for Computer Detection and Correction of Spelling Errors, *Comm. ACM*, **1964**, *7*, 171–176

[82] R. P. Murelli, A. K. Cheung, M. L. Snapper, Conformationally Restricted (+)-Cacospongionolide B Analogues. Influence on Secretory Phospholipase A_2 Inhibition, *J. Org. Chem.*, **2007**, *72*, 1545–1552

[83] http://www.rsc.org/Publishing/ReSourCe/AuthorGuidelines/ArticleLayout/sect3.asp

[84] Jeffrey E. F. Friedl, Mastering Regular Expressions, *O'Reilly and Associates* 2002

[85] L. P. Deutsch, B. W. Lampson, An online editor, *Comm. ACM*, **1967**, *10*, 793–799

[86] http://java.sun.com/javase/6/docs/api/java/util/regex/package-summary.html

[87] S. E. Adams, J. M. Goodman, R. J. Kidd, A. D. McNaught, P. Murray-Rust, F. R. Norton, J. A. Townsend, C. A. Waudby, Experimental data checker: better information for organic chemists, *Org. Biomol. Chem.*, **2004**, *2*, 3067–3070

[88] Private communication with J. Brazier and Dr J. Burton, Unilever Centre for Molecular Science Informatics 2005

[89] P. Wiklund, J. Bergman, Ring forming reaction of imines of 2-aminobenzaldehyde and related compounds, *Org. Biomol. Chem.*, **2003**, *1*, 367–372

[90] K. Hirota, K. Kazaoka, I. Niimoto, H. Sajiki, Efficient synthesis of 2,9-disubstitued 8-hydroxyadenine derivates, *Org. Biomol. Chem.*, **2003**, *1*, 1354–1365

[91] E. T. Gallagher, D. H. Grayson, Reactions of litiated (E)-3-halo-1-phenlssulfonylprop-1-enes and (Z)-1-halo-3-phenylsulfonylprop-1-enes with aldehydes, *Org. Biomol. Chem.*, **2003**, *1*, 1374–1381

[92] K. Smith, G. A. El-Hiti, A. J. Jayne, M. Butters, Acylation of aromatic ethers over solid acid catalysts: scope of the reaction with more complex acylating agents, *Org. Biomol. Chem.*, **2003**, *1*, 2321–2325

[93] X. Peng, N. Fukui, M. Mizuta, H. Suzuki, Nitration of moderately deactivated arenes with nitrogen dioxide and molecular oxygen under neutral conditions. Zeolite-induced enhancement of regioselectivity and reversal of isomer ratios, *Org. Biomol. Chem.*, **2003**, *1*, 2326–2335

[94] P. Wiklund, I. Romero, J. Bergman, Products from dehydration of dicarboxylic acids derived from anthranilic acid, *Org. Biomol. Chem.*, **2003**, *1*, 3396–3403

[95] F. Lecornué, J. Ollivier, Construction of medium-ring oxacycloalkenones. Extension towards benzo-fused cyclic ethers, *Org. Biomol. Chem.*, **2003**, *1*, 3600–3604

[96] F. Jeannot, G. Gosselin, C. Mathé, Synthesis and antiviral evaluation of 2'-deoxy-2'-C-trifluoromethyl b-D-ribonucleoside analogues bearing the five naturally occurring nucleic acid bases, Submitted for publication in *Org. Biomol. Chem.*

[97] R. Hunter, P. Richards, Stereoselective tetrapyrido[2,1-a]isoindolone synthesis via carbanionic and radical intermediates: a model study for the Tacaman alkaloid D/E ring fusion, Submitted for publication in *Org. Biomol. Chem.*

[98] M. D. Toscano, M. Frederickson, D.P. Evans, J.R. Coggins, C. Abell, C. González-Bello, Design, synthesis and evaluation of bifunctional inhibitors of type II dehydroquinase, Submitted for publication in *Org. Biomol. Chem.*

[99] C. A. Waudby, Unpublished work 2006

[100] J. A. Townsend, S. E. Adams, C. A. Waudby, V. K. de Souza, J. M. Goodman, P. Murray-Rust, Chemical documents: machine understanding and automated information extraction, *Org. Biomol. Chem.*, **2004**, *2*, 3294–3300

[101] A. Vasserman, Identifying Chemical Names in Biomedical Text: An Investigation of the substring co-occurence based approaches, *Proceedings of the Student Research Workshop at HLT-NAACL* 2004

[102] http://www.cambridgesoft.com/software/ChemDraw

[103] P. M. Elliott, Translation of Chemical Nomenclature by Syntax Controlled Techniques. *MSc. Thesis, The Ohio State University* 1969

[104] E. Garfield, An Algorithm for Translating Chemical Names to Molecular Formulas, *PhD Thesis* 1962

[105] D.I. Cooke-Fox, G.H. Kirby, J.D. Rayner, Computer Translation of IUPAC Systematic Organic Chemical Nomenclature. 1. Introduction and Background to a Grammar-Based Approach, *J. Chem. Inf. Comput. Sci.*, **1989**, *29*, 101–105

[106] D.I. Cooke-Fox, G.H. Kirby, J.D. Rayner, Computer Translation of IUPAC Systematic Organic Chemical Nomenclature. 2. Development of a Formal Grammar, *J. Chem. Inf. Comput. Sci.*, **106**, *29*, 106–112

[107] D.I. Cooke-Fox, G.H. Kirby, J.D. Rayner, Computer Translation of IUPAC Systematic Organic Chemical Nomenclature. 3. Syntax Analysis and Semantic Processing, *J. Chem. Inf. Comput. Sci.*, **1989**, *29*, 112–118

[108] D.I. Cooke-Fox, G.H. Kirby, J.D. Rayner, Computer Translation of IUPAC Systematic Organic Chemical Nomenclature. 4. Concise connection tables to structure diagrams, *J. Chem. Inf. Comput. Sci.*, **1990**, *30*, 122–127

[109] D.I. Cooke-Fox, G.H. Kirby, J.D. Rayner, Computer Translation of IUPAC Systematic Organic Chemical Nomenclature. 5. Steroid nomenclature Steroid nomenclature, *J. Chem. Inf. Comput. Sci.*, **1990**, *30*, 128–132

[110] G.H. Kirby, M.R.Lord, J.D. Rayner, Computer Translation of IUPAC Systematic Organic Chemical Nomenclature., 6. (Semi)automatic Name correction, *J. Chem. Inf. Comput. Sci.*, **1991**, *31*, 153–160

[111] A Guide to IUPAC Nomenclature of Organic Chemistry, Recommendations 1993, (including Revisions, Published and hitherto Unpublished, to the 1979 Edition of Nomenclature of Organic Chemistry), *IUPAC* 1993

[112] D. Weininger, SMILES, a Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules, *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 31–36

[113] D. Weininger, A. Weininger, J. L. Weininger, SMILES. 2. Algorithm for Generation of Unique SMILES Notation, *J. Chem. Inf. Comput. Sci.* **1989**, *29* 97–101

[114] D. Weininger, SMILES. 3. DEPICT. Graphical Depiction of Chemical Structures, *J. Chem. Inf. Comput. Sci.* **1990**, *30*, 237–243

[115] A. Copestake, M. A. Parker, S. Teufel, P. Murray-Rust, Extracting the Science from Scientific Publications, *EPSRC*, EP/C010035/1

[116] P. Corbett, P. Murray-Rust, High-throughput identification of chemistry in life science texts, *Computational Life Sciences II, Proceedings Lecture Notes in Computer Science*, **2006**, *4216*, 107–118

[117] P. Corbett, Unpublished work 2006

[118] A. Copestake, P. Corbett, P. Murray-Rust, C.J. Rupp, A. Siddharthan, S. Teufel, B. Waldron, An Architecture for Language Processing for Scientific Texts, *Proceedings of the UK e-Science All Hands Conference* 2006

[119] J. Brecher, Name=Struct: A Practical Approach to the Sorry State of Real-Life Chemical Nomenclature, *J. Chem. Inf. Comput. Sci.*, **1999**, *39* 943–950

[120] A. V. Aho, R. Sethi, J. D. Ullman, Compilers Principles, Techniques, and Tools, *Prentice-Hall* 2003

[121] P. Naur, Revised Report on the Algorithmic Language ALGOL 60, *Comm. ACM*, **1960**, *3*, 299–314

[122] M. Marcotty, H. Ledgard, The World of Programming Languages, *Springer-Verlag* 1986

[123] J. W. Backus, The Syntax and Semantics of the Proposed International Algebraic Language of the Zürich ACM-GAMM Conference, ICIP Paris, 1959

[124] E. L. Post, Formal Reductions of the General Combinatorial Decision Problem, *Am. J. Mathematics*, **1943**, *65*, 197–215

[125] http://www.pcre.org

[126] A. K. McCallum, Reinforcement Learning with Selective Perception and Hidden State, *PhD Thesis* 1995

[127] http://jedlik.phy.bme.hu/˜gerjanos/HMM/node4.html

[128] P. C. Austin, L. J. Brunner, J. E. Hux, Bayeswatch: an overview of Bayesian statistics, *J. Eval. Clin. Pract.*, **2002**, *8*, 277–286

[129] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Rubin, Bayesian Data Analysis, *Chapman & Hall* 1995

[130] http://research.microsoft.com/nlp

[131] N. Chomsky, Syntactic Structures, *Walter de Gruyter* 1957

[132] http://jflex.de

[133] B. W. Kernighan, D. M. Ritchie, The C programming Language, *Prentice-Hall* 1988

[134] J. R. Levine, T. Mason, D. Brown, Lex & Yacc, *O'Reilly & Associates* 1992

[135] S. E. Hudson, CUP LALR Parser Generator for Java, http://www.cs.princeton.edu/˜appel/modern/java/CUP/

[136] S. C. Johnson, YACC — Yet Another Compiler Compiler, *CS Technical Report Bell Telephone Laboratories*, **1975**, *32*

[137] E. Lindahl, B. Hess, D. van der Spoel, GROMACS 3.0: a package for molecular simulation and trajectory analysis, *J. Mol. Mod.*, **2001**, *7*, 306–317

[138] J. J. P. Stewart, MOPAC: A semiempirical molecular orbital program *J. Comput. Aided Mol. Des.*, **1990**, *4*, 1–45

[139] Private communication with Andrew Walkingshaw, Unilever Centre for Molecular Science Informatics 2007

[140] J. M. Soler, E. Artacho, J. D. Gale, A. Garca, J. Junquera, P. Ordejón and D. Sánchez-Portal, The SIESTA method for *ab initio* order-N materials simulation, *J. Phys. Condens. Matter*, **2002** , *14*, 2745–2779

[141] J. D. Gale, GULP: A computer program for the symmetry adapted simulation of solids, *J. Chem. Soc. Faraday Trans.*, 1997, 93, 629–637

[142] I. T. Todorov, W. Smith, DL_POLY_3: the CCP5 National UK Code for molecular-dynamics simulations, *Phil. Trans.*, **2004**, *362*, 1835–1852

[143] A. García, P. Murray-Rust, J. Wakelin, The use of XML and CML in Computational Chemistry and Physics Programs, *Proceedings of UK e-Science All Hands Conference*, 2004

[144] Private communication with Dr M. Braendle, Infozentrum Chemie Biologie Pharmazie 2005

[145] Private communication with Dr R. Kanters, Department of Chemistry, University of Richmond 2005

[146] Private communication with M. Howard, Jmol project 2005

[147] M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, J. A. Montgomery, Jr., T. Vreven, K. N. Kudin, J. C. Burant, J. M. Millam, S. S. Iyengar, J. Tomasi, V. Barone, B. Mennucci, M. Cossi, G. Scalmani, N. Rega, G. A. Petersson, H. Nakatsuji, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, M. Klene, X. Li, J. E. Knox, H. P. Hratchian, J. B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, P. Y. Ayala, K. Morokuma, G. A. Voth, P. Salvador, J. J. Dannenberg, V. G. Zakrzewski, S. Dapprich, A. D. Daniels, M. C. Strain, O. Farkas, D. K. Malick, A. D. Rabuck,

K. Raghavachari, J. B. Foresman, J. V. Ortiz, Q. Cui, A. G. Baboul, S. Clifford, J. Cioslowski, B. B. Stefanov, G. Liu, A. Liashenko, P. Piskorz, I. Komaromi, R. L. Martin, D. J. Fox, T. Keith, M. A. Al-Laham, C. Y. Peng, A. Nanayakkara, M. Challacombe, P. M. W. Gill, B. Johnson, W. Chen, M. W. Wong, C. Gonzalez, J. A. Pople, Gaussian 03, *Gaussian, Inc., Wallingford, CT*, 2004

[148] http://www.seti.org

[149] W. G. Richards, Virtual screening using grid computing: the screensaver project, *Nat. Rev. Drug Discovery*, **2002**, *1*, 551–555

[150] J. Basney, M. Livny, T. Tannenbaum, High Throughput Computing with Condor, *HPCU news*, *1*, (1997)

[151] M. W. Mutka, M. Livny, Scheduling remote processing capacity in a workstation-processor bank network, *Proc. 7th Int. Conf. Distributed Comput. Syst.*, **1987**, 2–9

[152] M. J. Litzkow, M. Livny, M. W. Mutka, Condor — a hunter of idle workstations, *Proc. 8th Int. Conf. Distributed Computing Systems*, **1988**, 104–111

[153] http://cactus.nci.nih.gov/ncidb2/download.html

[154] Y. Zhang, R. C. Glen, P. Murray-Rust, H. S. Rzepa, J. A. Townsend, Semantic grid computing — The WorldWideMolecularMatrix, *Abstr. Am. Chem. Soc.* 2004

[155] http://openbabel.sourceforge.net

[156] http://xml.apache.org/xindice

[157] http://wwmm.ch.cam.ac.uk/inchifaq

[158] S. J. Coles, N. E. Day, P. Murray-Rust, H. S. Rzepa, Y. Zhang, Enhancement of the chemical semantic web through the use of InChI identifiers, *Org. Biomol. Chem.*, **2005**, *3*, 1832–1834

[159] M. W. Schmidt, K. K. Baldridge, J. A. Boatz, S. T. Elbert, M. S. Gordon, J. H. Jensen, S. Koseki, N. Matsunaga, K. A. Nguyen, S. J. Su, T. L. Windus, M. Dupuis, J. A. Montgomery, General Atomic and Molecular Electronic Structure System, *J. Comput. Chem.*, **1993**, *14*, 1347–1363

[160] K. K. Irikura, D. J. Frurip, Computational Thermochemistry: Prediction and Estimation of Molecular Thermodynamics, *ACS Symp. Ser.*, 1998

[161] http://www.msg.ameslab.gov/GAMESS/GAMESS_Manual

[162] Private communication with Prof. K. K. Baldridge, University of Zürich, 2005

[163] J. Baker, A. Kessi, B. Delley, The generation and use of delocalized internal coordinates in geometry optimization, *J. Chem. Phys.*, **1996**, *105*, 192–212

[164] Private communtication with Dr W. Sudholt, University of Zürich, 2004

[165] Molecular Operating Environment, Version 2000.03, Chemical Computing Group Inc.

[166] R. C. Glen, A. Bender, C. H. Arnby, L. Carlsson, S. Boyer, J. Smith, Circular fingerprints: Flexible molecular descriptors with applications from physical chemistry to ADME, *IDrugs*, **2006**, *9*, 199–204.

[167] O. Ludwig, H. Schinke, W. Brandt, Reparametrisation of Force Constants in MOPAC 6.0/7.0 for Better Description of the Activation Barrier of Peptide Bond Rotations, *J. Mol. Mod.*, **1996**, *2*, 341–350

[168] P. P. Ewald, Fifty Years of X-Ray Diffraction, *Springer* 1962

[169] H. M. Rietveld, A profile refinement method for nuclear and magnetic structures, *J. Appl. Cryst.*, **1969**, *2*, 65–71.

[170] F. H. Allen, The Cambridge Structural Database: a quarter of a million crystal structures and rising, *Acta Cryst.*, **2002**, *B58*, 380–388

[171] W. L. Bragg, The Diffraction of Short Electromagnetic Waves by a Crystal, *Proc. Camb. Philos. Soc.*, **1912**, *17*, 43-57

[172] W. Clegg, A. J. Blake, R. O. Gould, P. Main, Crystal Structure Analysis: Principles and Practice, *Oxford University Press* 2002

[173] W. Friedrich, P. Knipping, M. Laue, Interferenzerscheinungen bei Röntgenstrahlen, *Annalen der Physik*, **1913**, *346*, 971–988

[174] P. P. Ewald, Zur Theorie der Interferenzen der Röntgentstrahlen in Kristallen, *Physik. Z.*, **1913**, *14*, 465–472

[175] D. Watkin, U_{equiv}: its past, present and future, *Acta. Cryst.*, **2000**, *B56*, 747–749

[176] F. L. Hirshfeld, Can X-ray data distinguish bonding effects from vibrational smearing?, *Acta Cryst.*, **1976**, *A32*, 239–244

[177] H. G. von Schnering, D. Vu, Are the Previously Described $[ClF_6][CuF_4]$ and $[Cu(H_2O)][SiF_6]$ Identical?, *Angew. Chem. Int. Ed.*, **1983**, *22*, 408

[178] B. Dittrich, T. Koritsánszky, P. Luger, A Simple Approach to Nonspherical Electron Densities by Using Invarioms, *Angew. Chem. Int. Ed.*, **2004**, *43*, 2718-2721

[179] N. K. Hansen, P. Coppens, Testing Aspherical Atom Refinements on Small-Molecule Data Sets, *Acta Cryst.*, **1978**, *A34*, 909–921

[180] B. Dittrich, C. B. Hübschle, M. Messerschmidt, R. Kalinowski, D. Girnta, P. Lugera, The invariom model and its application: refinement of D,L-serine at different temperatures and resolution, *Acta Cryst.*, **2005**, *A61*, 314–320

[181] B. Dittrich, P. Munshi, M. A. Spackman, Invariom-model refinement of L-valinol, *Acta Cryst.*, **2006**, *C62*, 633–635

[182] S. R. Hall, F. H. Allen, I. D. Brown, The Crystallographic Information File (CIF): a New Standard Archive File for Crystallography, *Acta Cryst.*, **1991**, *A47*, 655–685

[183] http://www.iucr.org/iucr-top/cif/cif_core/index.html

[184] S. R. Hall, The STAR file: a new format for electronic data transfer and archiving, *J. Chem. Inf. Comput. Sci.*, **1991**, *31*, 326–333

[185] S. R. Hall, N. Spadaccinit, The STAR File: Detailed Specifications, *J. Chem. Inf. Comput. Sci.*, **1994**, *34*, 505–508

[186] F. H. Allen, O. Kennard, W. D. S. Motherwell, W. G. Town, D. G. Waston, T. J. Scott, A. C. Larson, The Cambridge Crystallographic Data Centre. Part 3. The Unique Molecule Program, *J. Appl. Cryst.*, **1974**,*7*, 73–78

[187] http://checkcif.iucr.org

[188] http://journals.iucr.org/services/cif/datavalidation.html

[189] N. E. Day, P. Murray-Rust, H. S. Rzepa, S. M. Tyrrell, Y. Zhang, Automatic aggregation of open chemical data, *Abstr. Am. Chem. Soc.* 2005

[190] R. Guha, M. T. Howard, G. R. Hutchison, P. Murray-Rust, H. Rzepa, C. Steinbeck, J. Wegner, E. L. Willighagen, The Blue Obelisk — Interoperability in Chemical Informatics, *J. Chem. Inf. Model.*, **2006**, *46*, 991–998

[191] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, P. Li, Taverna: a tool for the composition and enactment of bioinformatics workflows, *Bioinformatics*, **2004**, *20*, 3045–3054

[192] Y. Zhang, P. Murrary-Rust, M. T. Dove, R. C. Glen, H. S. Rzepa, J. A. Townsend, S. Tyrrell, J. Wakelin, E. L. Willighagen, JUMBO –

An XML Infrastructure for eScience, *Proceedings of UK e-Science All Hands Conference*, 2004

[193] http://www.ccdc.cam.ac.uk/products/csd/request/request.php4

[194] http://journals.iucr.org/services/cif/reqditems.html

[195] P. Murray-Rust, H. S. Rzepa, S. M. Tyrrell, Y. Zhang, Representation and use of chemistry in the global electronic age, *Org. Biomol. Chem.*, **2004**, *2*, 3192–3203

[196] P. Murray-Rust, S. Tyrrell, CIF2CML — Automatic Processing in XML/CML, *Acta Cryst.*, **2005**, *A61*, C109

[197] http://ant.apache.org

[198] M. Calleja, B. Beckles, M. Keegan, M. A. Hayes, A. Parker, M. T. Dove, CamGrid: Experiences in constructing a university-wide, Condor-based, grid at the University of Cambridge, *Proceedings of UK e-Science All Hands Conference*, 2004

[199] Private communication with Dr C. Bolton, Unilever Centre for Molecular Science Informatics

[200] http://www.dspace.org

[201] L. Zorina, S. Khasanov, B. Narymbetov, R. Shibaeva, A. Kotov, É. Yagubskii, Crystal structure of radical cation salt, $(\mathit{BEDT\text{-}TTF})_4(GaCl_4)_2\ C_6H_5CH_3$, **2001**, *46*, 219-224

[202] Private communication with Dr Alison Edwards, Bragg Institute 2006

[203] Private communication with Dr W. Bernd Schweizer, Eidengenössische Technische Houchschule Zürich 2006

[204] http://www.ccdc.cam.ac.uk/products/csd/statistics

[205] http://www.iucr.org/iucr-top/cif/cifdic_html/1/cif_core.dic/Cchemical_formula.html

[206] X.-H., Li, H.-P. Xiao, *catena*-Poly[[[(2,2'-bipyridine)copper(II)]-μ-terephthalato] *N*,*N*-dimethylformamide solvate], *Acta Cryst.*, **2004**, *E60*, 898–900

[207] S. S. Shapiro, M. B. Wilk, An analysis of variance test for normality (complete samples), *Biometrika*, **1956**, *52*, 591–611

[208] C. R. Newton, I. S. Michela, G. W. J. Fleet, Y. Blériot, D. J. Watkin, 2-C-Hydroxymethyl-2,3-O-isopropylidene-D-ribono-1,5-lactam, *Acta Cryst.*, **2004**, *E60*, 909–910

[209] A.L. Spek, Single-crystal structure validation with the program PLATON, *J. Appl. Cryst.*, **2003**, *36*, 7–13

[210] N. E. Day, Unpublished work 2006

[211] S.F. Boys, Electronic wavefunctions. I. A general method of calculation for stationary states of any molecular system. *Proc. Roy. Soc.*, **1950**, *200*, 542–554

[212] J. C. Slater, A Simplification of the Hartree-Fock Method, *Phys. Rev.*, **1951**, *81*, 385–390

[213] P. A. M. Dirac, Note on exchange phenomena in the Thomas-Fermi atom, *Proc. Camb. Philos. Soc.*, **1930**, *26*, 376–385

[214] W. Kohn, L. J. Sham, Self-Consistent Equations Including Exchange and Correlation Effects, *Phys. Rev.*, **1965**, *A140*, 1133–1138

[215] A. D. Becke, Density-functional thermochemistry. III. The role of exact exchange, *J. Chem. Phys.*, **1993**, *98*, 5648–5652

[216] C. Lee, W. Yang, R. G. Parr, Development of the Colle-Salvetti correlation-energy formula into a functional of the electron density, *Phys. Rev.*, **1988**, *B37*, 785–789

[217] P. J. Stephens, F. J. Devlin, C. F. Chabalowski, M. J. Frisch, *Ab initio* calculation of vibrational absorption and circular dichroism spectra

using density functional force fields, *J. Chem. Phys.* **1994**, *98* 11623–11627

[218] J. Pople, D. L. Beveridge, P. A. Dobosh, Approximate Self-Consistent Molecular-Orbital Theory. V. Intermediate Neglect of Differential Overlap, *J. Chem. Phys.*, **1967**, *47*, 2026–2033

[219] M.J.S. Dewar, E. G. Zoebisch, E.F. Healy, J.J.P. Stewart, AM1: A New General Purpose Quantum Mechanical Molecular Model, *J. Am. Chem. Soc.*, **1985**, *107*, 3902–3909

[220] J. J. P. Stewart, Optimization of parameters for semiempirical methods .1. Method, *J. Comp. Chem.*, **1989**, *10* 209–220

[221] J. J. P. Stewart, Optimization of parameters for semiempirical methods .2. Applications, *J. Comp. Chem.*, **1989**, *10* 221–264

[222] http://java.sun.com/j2se/1.5.0/docs/api

[223] http://alpha2.bmc.uu.se

References are given in the style adopted by the Journal of Chemical Informatics and Computer Science.