

# Automated Analysis and Validation of Open Chemical Data

Nicholas Elliot Day

Emmanuel 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.

November 20, 2008

## 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  $\text{\LaTeX}2\epsilon$  according to the specifications defined by the Board of Graduate Studies and the Chemistry Degree Committee.

## Abstract

Methods to automatically extract Open Data from the chemical literature, validate it, and use it to validate theory are examined.

Chemical identifiers which assist the automatic location of chemical structures using commercial Web search engines are investigated. The IUPAC International Chemical Identifier (InChI) gives almost 100% recall and precision, though is shown to be too long for present search engines. A combination of InChI and InChIKey, a shorter, fixed-length hash of the InChI string, is concluded to be the best current method of identifying structures.

The proportion of published, Open Crystallographic Information Files (CIFs) that are valid with respect to the specification is shown to be improving, and is around 99% in 2007. The error rate in the conversion of valid CIFs to Chemical Markup Language (CML) is less than 0.2%. The machine generation of connection tables from CIFs requires many heuristics, and in some cases it is impossible to deduce the exact connection table.

CrystalEye, a fully-automated system for the reformulation of the fragmented crystallographic Web into a structured XML-based repository is described. Published, Open CIFs can be located and aggregated programmatically with almost 100% recall. It is shown that, by converting CIF data to CML, software can be created to use the latest Web standards and technologies to enhance the ability of Web users to browse, find, keep updated, download and reuse the latest published crystallography.

A workflow for the high-throughput calculation of solid-state geometry using a semi-empirical method is described. A wide-range of organic and inorganic systems provided by CrystalEye are used to test both the data and the method. Several errors in the method are discovered, many of which can be attributed to the parameterization process.

An Open NMR experiment to perform high-throughput prediction of  $^{13}\text{C}$  chemical shifts using a GIAO protocol is described. The data and analysis were provided on publicly-available webpages to enable *crowdsourcing*, which assisted in discovering an error rate of 6.1% in the starting data. The protocol was refined during the work and shown to have an average unsigned error of 2.24ppm for  $^{13}\text{C}$  nuclei of small, rigid molecules; comparable to the errors observed elsewhere for general structures using HOSE and Neural Network methods.

## Acknowledgments

I would like to thank Dr Peter Murray-Rust for his advice and guidance throughout this work. Dr Joe Townsend, Jim Downing, Dr Yong Zhang, Dr Andrew Walkingshaw, Dr Nico Adams and Dr Simon Tyrrell are thanked for useful discussions and advice throughout the course of my PhD. I would also like to thank the UCC computing staff, particularly Dr Charlotte Bolton, without which this work would not have been possible. The EPSRC is thanked for funding.

Cheers also to Dan, Andy, Rob and Jon for providing copious amounts of tea and distracting ‘activities’. As always, thanks most to Mum, Dad, Vick and Anna.

# Contents

<b>Disclaimer</b>	<b>i</b>
<b>Abstract</b>	<b>ii</b>
<b>Acknowledgements</b>	<b>iv</b>
<b>Table of contents</b>	<b>viii</b>
<b>List of tables</b>	<b>ix</b>
<b>List of figures</b>	<b>xv</b>
<b>Glossary</b>	<b>xvi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Chemical identification . . . . .	2
1.2 Open Data . . . . .	2
1.3 Data and Metadata . . . . .	4
1.4 Machine-understandable chemical data . . . . .	5
1.4.1 The Semantic Web . . . . .	6
1.5 Open Source software . . . . .	8
1.6 e-Science . . . . .	8
1.7 Aims . . . . .	9
<b>2 Locating chemical data on the Web</b>	<b>11</b>
2.1 Introduction . . . . .	11
2.2 Web search-engines . . . . .	11
2.2.1 Sitemaps . . . . .	12
2.3 Representing chemical entities as strings . . . . .	13
2.4 The IUPAC International Chemical Identifier . . . . .	16
2.4.1 InChI creation and structure . . . . .	17
2.4.2 Searching for InChIs on the Web . . . . .	22
2.4.3 The Google-InChI Web Service . . . . .	25
2.4.4 Staurosporine - an InChI case study . . . . .	26

2.5	Conclusions . . . . .	38
<b>3</b>	<b>Creating and deriving data in CML from CIF</b>	<b>39</b>
3.1	Introduction . . . . .	39
3.2	Why convert CIF to CML? . . . . .	40
3.3	The Crystallographic Information Framework . . . . .	42
3.3.1	STAR file concepts and syntax . . . . .	42
3.3.2	CIF syntax . . . . .	44
3.3.3	CIF dictionaries . . . . .	46
3.3.4	checkCIF . . . . .	47
3.4	CIFXML: converting CIF to XML . . . . .	49
3.4.1	Conformance to CIF . . . . .	52
3.4.2	CIFXML-J functionality . . . . .	53
3.4.3	Representation of CIFs in XML . . . . .	54
3.4.4	CIFXML-J architecture . . . . .	57
3.4.5	Using CIFXML-J . . . . .	60
3.4.6	CIF to CIFXML conversion statistics . . . . .	64
3.5	CIFConverter: converting CIF to CML . . . . .	67
3.5.1	Handling multiple datablocks . . . . .	72
3.5.2	Building the CML . . . . .	72
3.5.3	Adding symmetry elements . . . . .	76
3.5.4	Dictionary validation . . . . .	76
3.5.5	Using CIFConverter . . . . .	78
3.5.6	CIFXML to CML conversion statistics . . . . .	80
3.6	Enhancing the CML . . . . .	82
3.6.1	Creating the connection table . . . . .	84
3.6.2	Adding canonical identifiers . . . . .	98
3.7	Current usage . . . . .	99
3.8	Conclusions . . . . .	99
<b>4</b>	<b>Automating the aggregation, creation and dissemination of semantic crystallography</b>	<b>101</b>
4.1	Introduction . . . . .	101
4.2	Implementing a workflow system . . . . .	103
4.2.1	The Taverna Workbench . . . . .	103
4.2.2	Component development and workflows . . . . .	107
4.2.3	Developing under Taverna . . . . .	111
4.3	CrystalEye . . . . .	114
4.3.1	Implementation . . . . .	116
4.3.2	Aggregation . . . . .	116
4.3.3	Processing the data . . . . .	129
4.3.4	The website . . . . .	134
4.3.5	Services . . . . .	140

4.3.6	Database distribution . . . . .	151
4.3.7	CrystalEye data in RDF . . . . .	156
4.3.8	Further work . . . . .	158
4.4	Conclusions . . . . .	160
<b>5</b>	<b>High-throughput prediction of solid-state geometry using semi-empirical methods</b>	<b>161</b>
5.1	Introduction . . . . .	161
5.2	MOPAC-CML . . . . .	163
5.2.1	FoX . . . . .	163
5.2.2	MOPAC and FoX . . . . .	165
5.2.3	MOPAC-CML and Jmol . . . . .	167
5.3	High-throughput computing . . . . .	167
5.3.1	Condor . . . . .	170
5.3.2	CamGrid . . . . .	170
5.4	The calculation workflow . . . . .	171
5.4.1	Structure selection . . . . .	171
5.4.2	MOPAC input . . . . .	172
5.4.3	Condor input . . . . .	175
5.4.4	Job submission and retrieval . . . . .	177
5.4.5	The overall workflow . . . . .	178
5.5	The first protocol . . . . .	178
5.5.1	Organic structures . . . . .	181
5.5.2	Inorganic structures . . . . .	182
5.6	The second protocol . . . . .	184
5.6.1	The data . . . . .	186
5.6.2	Inorganic structures . . . . .	187
5.6.3	Organic structures . . . . .	198
5.7	Conclusions . . . . .	212
<b>6</b>	<b>High-throughput prediction of <math>^{13}\text{C}</math> NMR chemical shifts by quantum-mechanical GIAO calculations</b>	<b>215</b>
6.1	Introduction . . . . .	215
6.2	Nuclear Magnetic Resonance . . . . .	215
6.3	Computational NMR . . . . .	216
6.3.1	The Rzepa Protocol . . . . .	218
6.3.2	NMRShiftDB . . . . .	219
6.4	Calculations . . . . .	222
6.4.1	Structure selection . . . . .	222
6.4.2	Gaussian03 input . . . . .	223
6.4.3	Condor input . . . . .	224
6.4.4	TMS . . . . .	225
6.5	Open Computational NMR . . . . .	226



6.6	Analysis . . . . .	227
6.6.1	Preparing the output . . . . .	227
6.6.2	Initial results . . . . .	229
6.6.3	Sources of error . . . . .	230
6.6.4	Cleaning the dataset . . . . .	233
6.6.5	Conformational issues . . . . .	237
6.6.6	HSR1 . . . . .	239
6.6.7	Conclusions . . . . .	244
<b>7</b>	<b>Conclusions</b>	<b>246</b>
<b>A</b>	<b>Analysis of the efficacy of Web search engines for chemical search</b>	<b>252</b>
A.1	Search-engines and strategy . . . . .	252
A.2	Search terms and metrics . . . . .	256
A.3	Searching for CAS numbers . . . . .	257
A.4	Searching for InChIs . . . . .	259
A.4.1	The InChI architecture and implications . . . . .	259
A.4.2	Results . . . . .	260
A.5	Searching for SMILES . . . . .	261
A.6	Searching for InChI strings from the KEGG collection using Google . . . . .	262
<b>B</b>	<b>MOPAC calculation references</b>	<b>270</b>
B.1	Second-protocol inorganic calculations . . . . .	270
B.1.1	Short atom-atom distances . . . . .	270
B.1.2	Silicas . . . . .	271
B.1.3	Errors in the data . . . . .	272
B.1.4	Errors in modeling . . . . .	272
B.2	Organic calculations . . . . .	273
B.2.1	Calculations that terminated with controlled errors . . . . .	273
B.2.2	Calculations containing radicals that converged successfully . . . . .	276
B.2.3	Changes in connection table . . . . .	276
B.2.4	Density change outliers . . . . .	277
<b>C</b>	<b>Published Work</b>	<b>282</b>
	<b>Bibliography</b>	<b>284</b>

# List of Tables

3.1	Example of applying the canonicalization algorithm . . . . .	61
3.2	Proportion of error occurrences in CIF to CML conversion . .	82
5.1	Table showing the density change mean and standard deviation for structures with various minimum translation vectors. .	211
6.1	Comparison of the linear fitting statistics for HSR0 and HSR1	242
6.2	Linear fitting coefficients and average absolute shift deviation using only those spectra which were taken at fields of over 25Hz	244
A.1	Selection of search engines used in the analysis . . . . .	252
A.2	Number of query results viewable for a selection of commercial Web search engines . . . . .	256
A.3	Searching for CAS numbers with various strings . . . . .	258
A.4	Recall of InChI strings from the Crystal Structure Report Archive . . . . .	261
A.5	Searching for SMILES representations of caffeine in Web search engines . . . . .	262
A.6	Collisions in web queries for InChI representations of Kegg molecules . . . . .	263
B.1	The structures for the major components of the 18 calculations that threw the “all convergers are now forced on” error and didn’t have bad starting geometry . . . . .	274
B.2	The organic structures which were predicted to have a density change of <-20% . . . . .	277
B.3	The organic structures which were predicted to have a density change of >20% . . . . .	279

# List of Figures

1.1	The Semantic Web Stack . . . . .	7
2.1	Different representations of the same compound have the same InChI . . . . .	18
2.2	The classes of structural information that can currently be represented in an InChI . . . . .	19
2.3	The InChI layer structure allows chemical entities to be represented to the known level of detail . . . . .	21
2.4	The InChI representation of caffeine . . . . .	21
2.5	GDP- <i>D</i> -mannose (top) and GDP- <i>L</i> -mannose and their corresponding InChI representations . . . . .	23
2.6	The Google-InChI webpage with naphthalene drawn in the structure editor. . . . .	27
2.7	The results page for the Google-InChI service after naphthalene was submitted. . . . .	28
2.8	Structure of Staurosporine in 2D showing absolute stereochemistry . . . . .	29
2.9	Conformation of Staurosporine in the solid state . . . . .	30
2.10	Representations of Staurosporine found on the Web with no stereochemical information provided . . . . .	31
2.11	Representations of Staurosporine found on the Web with some stereocentres defined, but incorrectly . . . . .	31
2.12	Representations of Staurosporine found on the Web with all stereocentres defined, but incorrectly . . . . .	32
2.13	Representations of Staurosporine found on the Web with incorrect connectivity . . . . .	33
2.14	Representations of Staurosporine found on the Web with impossible stereochemistry . . . . .	33
2.15	Staurosporine as it appears in the wInChI program . . . . .	34
3.1	Comparison between the Crystallographic Information Framework and XML . . . . .	41
3.2	Example STAR file data block . . . . .	44
3.3	Example data name definition in the coreCIF dictionary . . . . .	46

3.4	The checkCIF service homepage . . . . .	48
3.5	Alert section from a checkCIF report . . . . .	50
3.6	The CIFXML-J inheritance hierarchy . . . . .	59
3.7	Plot showing the percentage of failures in parsing CIFs to CIFXML using all CIFs aggregated by CrystalEye from 2001-7	65
3.8	Plot showing the percentage of failures in parsing CIFs to CIFXML using CIFs aggregated by CrystalEye from ACS, Acta Cryst. and RSC from 2001-7 . . . . .	66
3.9	Number of CIFXML-J parsing errors per year by cause for RSC CIFs . . . . .	68
3.10	Number of CIFXML-J parsing errors per year by cause for ACS CIFs . . . . .	69
3.11	CIFConverter's method of handling CIFs containing more than one structure datablock . . . . .	73
3.12	Plot showing the percentage of failures in converting CIFs to CML using all CIFs aggregated by CrystalEye from 2001-7 . .	81
3.13	Rendering of structural data contained within a CIF . . . . .	83
3.14	Disordered methyl group represented in CIF . . . . .	85
3.15	Rendering of the CML documents showing the removal of mi- nor disordered structural components . . . . .	86
3.16	Examples of published representations of invalid CIF disorder.	87
3.17	The creation of the complete unit cell for an inorganic crystal structure . . . . .	88
3.18	Images showing the creation of a complete molecular skeleton from non-disordered atom sites . . . . .	89
3.19	. . . . .	90
3.20	Finding the sets of bond orders and charges for a crystal with multiple moieties where no moiety charges have been provided	92
3.21	The process of adding bond orders and charges to a molecular skeleton. . . . .	93
3.22	Adding BOACs to an organometallic moiety where both the moiety charge and metal charge are known . . . . .	95
3.23	Attempt to add BOACs to a Copper-Terbium complex where no moiety or metal charges have been provided . . . . .	96
4.1	Article summary in the table of contents from Acta Crystal- lographica Section E . . . . .	102
4.2	The Taverna Workbench GUI . . . . .	106
4.3	Key for the workflow components as rendered in the Taverna GUI . . . . .	107
4.4	Examples of workflow data visualization offered by the enactor invocation window . . . . .	108
4.5	WWMM LocalWorkers in the available services panel in Taverna	109

4.6	WWMM WSs shown in the available services panel of Taverna	110
4.7	Workflow to find and aggregate CIFs from IUCr journals	112
4.8	Workflow to convert CIFs to enhanced CML	113
4.9	Enactor invocation window highlighting the lack of detail shown of the components of nested workflows	115
4.10	The route from a journal TOC to the CIFs for each publisher scraped by CrystalEye	120
4.11	The steps taken by the spider for RSC journals to find CIFs	121
4.12	Plot showing the number of CIFs aggregated by CrystalEye from publisher's websites between 1990 and 2007	125
4.13	Plot showing the number of CIFs aggregated by CrystalEye from each publisher from 1990 to 2007	126
4.14	Number of articles published in ACS journals from which over 500 CIFs have been aggregated between 2001-7	127
4.15	The number of CIFs provided per article in ACS journals from which over 500 CIFs have been aggregated between 2001-7	128
4.16	The relationship between checkCIF report HTML and check-CIF XML	131
4.17	Example 2D images generated from CrystalEye CML	134
4.18	Example of overlapping atoms and bonds in 2D structure generation in CrystalEye	135
4.19	Fragment types generated for each moiety in CrystalEye	136
4.20	Example of a ring-nucleus fragment that has been singly and doubly sprouted	136
4.21	Browsing the crystallography in CrystalEye by journal issue	137
4.22	Browsing the crystallography from an issue of Organic and Biomolecular Chemistry in CrystalEye	138
4.23	Full CrystalEye crystal summary webpage	139
4.24	CrystalEye search page	140
4.25	Example substructure search of CrystalEye	142
4.26	Example cell parameter search of CrystalEye	143
4.27	The different types of RSS and CMLRSS feed available in CrystalEye	144
4.28	RSS feed for structures containing carbon-silicon bonds	145
4.29	Image on the interactive histogram of the lengths of all nickel-nitrogen bonds in the structures in CrystalEye	149
4.30	List of links to the bond-length histograms for carbon	150
4.31	Example of the CrystalEye Greasemonkey script in action	152
4.32	Results page of an SPARQL query of bibliographic data from CrystalEye in RDF	158
5.1	MOPAC input file to calculate the solid-state geometry of Albite	164

5.2	MOPAC output file from the calculation of the solid-state geometry of Albite . . . . .	164
5.3	Section of a MOPAC-CML output file showing the structure geometry being output every 25th cycle . . . . .	168
5.4	Figure showing the use of Jmol to view the geometry of a structure at various points during a calculation from the MOPAC-CML output file. . . . .	169
5.5	Example input file for a MOPAC solid-state calculation . . . .	173
5.6	Example submit file for the submission of a MOPAC job to Condor . . . . .	176
5.7	The workflow for performing high-throughput solid-state MOPAC calculations . . . . .	179
5.8	Figure showing an experimentally observed charge-transfer complex of <i>N</i> -Iodosuccinimide and an imine . . . . .	180
5.9	Plot showing the experimentally observed densities against the change in density predicted by MOPAC for the set of converged organic structure calculations . . . . .	182
5.10	Plot showing the experimentally observed densities against the change in density predicted by MOPAC for the set of successfully converged inorganic structure calculations . . . . .	184
5.11	Plot showing the experimentally observed densities against the change in density predicted by MOPAC for the set of 1258 successfully converged inorganic structure calculations . . . . .	189
5.12	Figure showing the creation of a spurious Ca-Na bond in prober-tite . . . . .	190
5.13	Figure showing the starting cluster of potassium dinitramide displayed in Jmol . . . . .	191
5.14	Plot showing the experimentally observed densities against the change in density predicted by MOPAC for the set of 393 remaining inorganic structure calculations . . . . .	192
5.15	Density plot of the RMS atomic deviations for the 393 remaining calculations . . . . .	193
5.16	Figure showing the erroneous starting structure of SiF <sub>4</sub> containing F-F bonds . . . . .	195
5.17	Figure showing the large expansion in geometry predicted for NiF <sub>2</sub> . . . . .	197
5.18	Figure showing the large loss of symmetry predicted during the calculation for Ag <sub>2</sub> O . . . . .	198
5.19	Figure showing the data in the CIF for erucic acid rendered in Jmol . . . . .	200
5.20	Figure showing the formation of an N-I bond by contraction of the cell . . . . .	202
5.21	Figure showing the formation of an S-S bond . . . . .	203

5.22	Figure showing two structures in which the formation of a new S-N bond is predicted . . . . .	204
5.23	Plot of all observed O-I distances of less than 5Å against the calculated distance . . . . .	205
5.24	Plot of all observed N-Br distances of less than 5Å against the calculated distance . . . . .	206
5.25	Density plot of the RMS atomic deviations for the 4237 remaining calculations . . . . .	207
5.26	Figure showing the rotating geometry of $\alpha$ -(2-pyridine)-2,4-dinitrophenylethenyl at various stages during calculation . . .	208
5.27	Plot of the angle of rotation of each structure against its RMS deviation . . . . .	209
5.28	Plot of the minimum translation vector against the change in density during calculation for each structure . . . . .	210
6.1	Gaussian03 workflow implementing HSR0 . . . . .	219
6.2	Gaussian03 workflow template for HSR0 . . . . .	224
6.3	Condor submit file for Gaussian03 jobs . . . . .	225
6.4	Webpage showing an interactive graph linked to a Jmol applet of NMR calculation result . . . . .	228
6.5	Section of the output of a Gaussian03 GIAO-based calculation of NMR shifts . . . . .	229
6.6	Plot showing all calculated versus observed $^{13}\text{C}$ shifts for the 295 successfully completed calculations using HSR0 . . . . .	230
6.7	Plot showing all calculated versus observed $^{13}\text{C}$ shifts for the 295 successfully completed calculations using HSR0 after spin-orbit offsets have been applied . . . . .	231
6.8	Webpage showing an interactive graph of RMS deviation vs. mean deviation for all structures calculated with HSR0 after spin-orbit offsets have been applied . . . . .	235
6.9	Webpage showing an interactive graph of a ‘misassignment’ plot of difference between observed and calculated shifts against the average of the two . . . . .	236
6.10	Plot showing the shifts for chemically equivalent atoms being inequivalent due to the geometry of the optimized structure. .	238
6.11	Figure showing averaging of shifts for Morgan equivalent atoms in styrene . . . . .	238
6.12	Template for Gaussian03 input files implementing HSR0 . . .	240
6.13	Calculated vs. observed chemical shifts for the HSR1 protocol after spin-orbit offsets and averaging of the shifts for topologically equivalent atoms have been applied . . . . .	241
6.14	Calculated vs. observed shifts for HSR1 for those structures with spectra determined at a field of over 25Hz . . . . .	243

7.1	Screenshot of the C3DE application. . . . .	250
-----	---	-----



## Glossary

**API** Applicaton Programming Interface

**ACS** American Chemical Society

**CAS** Chemical Abstracts Service

**CCDC** Cambridge Crystallographic Data Centre

**CDK** Chemistry Development Kit

**CIF** Crystallographic Information File

**CML** Chemical Markup Language

**COD** Crystallography Open Database

**COMCIFS** Committee for the Maintenance of the CIF Standard

**CSD** Cambridge Structural Database

**CT** Connection Table

**DDL** Dictionary Definition Language

**DFT** Density Functional Theory

**DOI** Digital Object Identifier

**DOM** Document Object Model

**DTD** Document Type Definition

**GIAO** Gauge-Including Atomic Orbitals

**GUI** Graphical User Interface

**H-M** Hermann-Mauguin

**HOSE** Hierarchical Organisation of Spherical Environments

**HTML** HyperText Markup Language

**HTTP** HyperText Transfer Protocol

**ICSD** Inorganic Crystal Structure Database

**IDE** Integrated Development Environment

**IETF** Internet Engineering Task Force

**InChI** IUPAC International Chemical Identifier

**IUCr** International Union of Crystallography

**IUPAC** International Union of Pure and Applied Chemistry

**JAR** Java Archive

**MIME** Multipurpose Internet Mail Extension

**MM** Molecular Mechanics

**MSDS** Material Safety Data Sheets

**NLP** Natural Language Parsing

**NMR** Nuclear Magnetic Resonance

**NN** Neural Networks

**ONS** Open Notebook Science

**PM6** Parametric Method number 6

**QM** Quantum Mechanics

**RDF** Resource Description Framework

**RDFa** Resource Description Framework attributes

**REST** Representational State Transfer

**RHF** Restricted Hartree-Fock

**RMS** Root Mean Squared

**RSC** Royal Society of Chemistry

**RSS** Rich Site Summary

**RTECS** Registry of Toxic Effects of Chemical Substances

**SAX** Simple API for XML

**SMILES** Simplified Molecular Input Line Entry Specification

**SPARQL** SPARQL Protocol and RDF Query Language

**SPECTRa** Submission, Preservation and Exposure of Chemistry Teaching and Research Data

**STAR** Self-defining Text Archival and Retrieval

**StAX** Streaming API for XML

**STO** Slater-Type Orbital

**SVG** Scalable Vector Graphics

**TOC** Table of Contents

**TMS** Tetramethylsilane

**UCC** Unilever Centre for Molecular Informatics

**UHF** Unrestricted Hartree-Fock

**URL** Uniform Resource Locator

**VM** Virtual Machine

**W3C** World Wide Web Consortium

**WS** Web Service

**WSDL** Web Services Description Language

**WWMM** World-Wide Molecular Matrix

**XML** eXtensible Markup Language

**XSD** XML Schema Definition

# Chapter 1

## Introduction

Data is seen as an integral part of any scientific discipline, and the ability to view and analyse another scientist’s data is essential. Much classical chemical knowledge was discovered as a result of humans gathering data from published literature, *e.g.* the Periodic Table and the Bürgi-Dunitz angle[1]. There are huge amounts of undiscovered science in the scientific literature, and this thesis is aimed at examining if and how we can perform *data-driven science* using data contained in the current literature.

The introduction of computers into science has had a profound effect on the way data is created and gathered, giving rise to what has been termed as a *data deluge*[2]. Scientists can now produce much more data than ever feasible before, *e.g.* through mechanisation enabling automated high-throughput experimental procedures, or through faster and more widely distributed computers letting us perform more, and bigger, simulations. As a result, it is now common for the Chemical Abstracts Service (CAS) to report well over 1 million new structures per year[3]. It is likely that the volume of data generated in research and by scientific instruments will soon dwarf all the technical and scientific data collected in the history of research.

With such a large volume of data being produced, it is becoming less and less feasible to manually process and examine data to identify potentially interesting features and discover significant relationships. Instead, it is necessary to automate the process, though this is predicated on:

- the ability to programmatically identify all potentially interesting documents or items of data,
- the ability to gain automated access to data,
- that data items are provided with enough metadata to judge whether they are suitable and of sufficient quality,
- that data items be in a form that a machine can be taught to understand.

## 1.1 Chemical identification

Chemistry is based on the synthesis, structure and properties of molecules; therefore it is imperative that a molecule can be uniquely and globally identified. IUPAC nomenclature has long been the standard identification mechanism, and in principle it identifies chemical structures uniquely. However, in practice this is not the case, and there may also be many ways of representing the same structure (*e.g.* tautomers).

Human chemists are trained to recognise these implicit equivalences, but it is difficult to provide a computer with the same ability. Thus, it is important that another unique identifier is used which removes the need to provide machines with a high-level of chemical perception. Machine-understandable chemical identification and methods of locating chemical structures in electronic form are discussed further throughout chapter 2.

## 1.2 Open Data

The traditional method of disseminating chemical information has been through peer-reviewed literature. The majority of this, however, is Closed (*i.e.* behind toll or permission barriers), as are the collections of data abstracted by secondary publishers such as CAS. Machines cannot make payments or ask for permission for reuse of data, so to enable automated data-driven science, data must be made Open (*i.e.* where toll and some or all permission

barriers are removed) and provided with an explicit machine-understandable declaration stating its level of Openness.

Open Access, as defined in the Budapest[4], Bethesda[5] and Berlin[6] declarations, is the free, unrestricted access for download and reuse of data. The Open Access movement promotes making information Open through two mechanisms, deposition into Institutional Repositories (*e.g.* DSpace[7] or ePrints[8]), or by publishing in Open Access peer-reviewed literature. However, as currently implemented, the level of access provided by Open Access is often unclear, and commonly refers to the fact that the data is freely readable, but gives no permission for downloading and reuse. This has led to the definition of another term, Open Data, as:

... a philosophy and practice requiring that certain data are freely available to everyone, without restrictions from copyright, patents or other mechanisms of control.[9]

A number of Open Data licenses have been developed, including Science Commons[10] and Open Data Commons[11], which data providers can use to tag their data.

We now see several different steps being taken to provide Open Access data by publishers and institutions, for instance:

- the creation of fully Open Access journals(*e.g.* Nucleic Acids Research[12]),
- the conversion of Closed Access journals to fully Open Access (*e.g.* Acta Crystallographica Structure Reports[13]),
- Closed journals adopting a mixed-access model, where authors can pay a fee to make their article Open Access,
- mandates for Open Access to NIH-funded research, and for research performed at the universities of Harvard[14], Stirling and Southampton.

It is unclear how much of this Open Access data will be provided as Open Data, though the movement toward the former is clearly an important checkpoint toward the latter. The number of Open Data collections in chemistry is still small, though those provided by PubChem[15] and Wikipedia[16] are already being used for exciting projects (*e.g.* wiclopedia[17]). CrystalEye, a collection of Open crystallographic data was created during the work for this thesis, and is described in chapter 4.

### 1.3 Data and Metadata

Metadata is data about data. Examples of metadata concerning a scientific experiment might include the temperature and pressure at which the experiment was performed, who performed it and the equipment used. Metadata is vital to assess the suitability and quality of data, and hence vital to discovery within data and its subsequent exploitation.

The publication process in many areas of science forces a decoupling of the interpretation and analysis, which is published, from the data and metadata, which is often not published in full. It should now be possible to publish a fairly complete scientific record of an experiment, yet the current publication process continues to emphasize the article at the expense of the data. However, some communities insist on data being published, including much of bioscience (*e.g.* protein sequences are deposited with NCBI[18] and structures with PDB[19]) and crystallography.

Many journals now require that a CIF[21] (Crystallographic Information File) be provided as supplementary data to any article describing a crystal structure. These CIFs contain the complete output of a crystallographic experiment in a machine-understandable format, and thousands are now published each month, a large number of which are Open. Data and metadata contained within CIF files are used extensively throughout this thesis, and are discussed at length throughout chapters 3, 4 and 5.

## 1.4 Machine-understandable chemical data

When we refer data as being machine-understandable, we are stating that the data is in a format for which we can teach a machine to understand the following:

- **syntax** - how the document byte stream should be tokenized,
- **semantics** - the meaning that has been provided for tokens or larger components of documents.

Electronic chemical publications currently available are essentially electronic analogues of their paper counterparts, and are designed to be read solely by humans. Thus, they are not conducive for automated parsing, *e.g.* a text string such as:

Compound 2a melted at 119°C.

is simple for a human to understand, but requires much effort to allow a machine to extract the data items and their meaning, *i.e.* that 119°C is recognised as a temperature with units, and that it is associated with compound 2a. However, there are several ways of stating the same information using natural language, and creating software that would correctly parse each one is difficult. There are also other complexities such as how the reference to compound 2a would be associated to a chemical structure.

Natural language parsing (NLP) is a very difficult task, and is being investigated in chemistry in the SciBorg project[22]. While this work is vital to create data from historical chemical documents, it is important that NLP is not relied upon for the future extraction of data.

Providing data in formats such as CIF is an important step, but it is equally necessary to take advantage of the hyperlinking afforded by the Web. This way, articles and data can be published together in a coherent format, where there is no separation between the document and the data. Such



a composition has already been described as a *datument*[23]. A first step towards this has been taken by the RSC in their work on Project Prospect[24].

### 1.4.1 The Semantic Web

To assist the description and machine-extraction of data, we can take advantage of the technologies created for the Semantic Web (SW), which has been described as:

...an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.[25]

In the SW, all data is described using highly-structured markup, where layers of semantics are provided by successive standards. The SW comprises the following standards (which are organized in the Semantic Web Stack - see figure 1.1):

**XML** provides an elemental syntax for content structure within documents, yet associates no semantics with the meaning of the content contained within,

**XML Schema** a language for providing and restricting the structure and content of elements contained within XML documents,

**RDF** a language for expressing data models, which refer to objects (often web resources) and their relationships,

**RDF Schema** a vocabulary for describing properties and classes of RDF-based resources, with semantics for generalized-hierarchies of such properties and classes,

**OWL** adds more vocabulary for describing properties and classes.

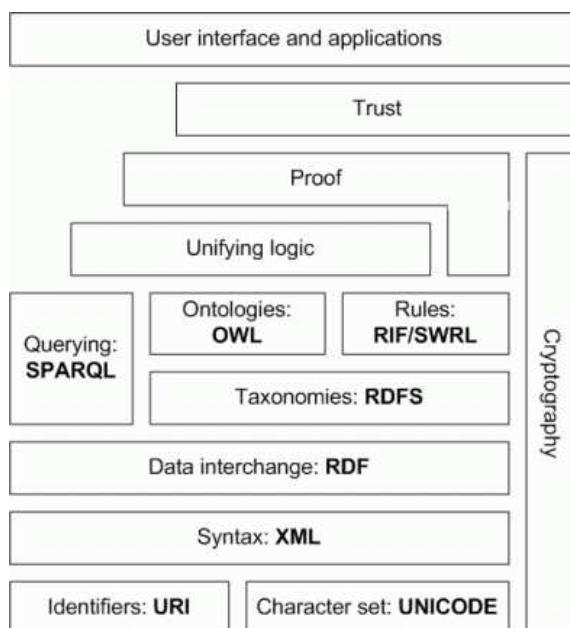


Figure 1.1: The Semantic Web Stack[26]. Note that standards and technologies have yet to be implemented for some aspects of the stack.

An XML language for chemistry, CML (Chemical Markup Language) has been described by Murray-Rust and Rzepa[27, 28]. The core of CML is aimed at describing molecules and their properties, though a number of modules have been developed to describe other aspects of chemistry, such as crystallography (CMLCryst) and spectra (CMLSpect[29]).

To highlight the power of CML for representing chemical data, we can markup the earlier natural language example with CML:

```
<cml:molecule ref='2a'>
  <cml:property>
    <cml:scalar dictRef='prop:mpt'
      units='units:celsius'
      dataType='xsd:float'
    >119</cml:scalar>
  </cml:property>
```

`</cml:molecule>`

Here, the data is surrounded by CML tags, enabling easy programmatic access to each item, while the addition of semantics is enabled through tag names and references to dictionaries in element attributes.

A benefit afforded by using SW protocols to represent data is that we can use Open Source (see below) tools created by others for manipulation, querying and visualization of data in these formats in our own work. It is common for these tools to have large user and developer communities, and as a result are robust and have many useful features. For instance **XOM**, a popular Open Source library for reading, manipulating, querying and writing XML documents, is used as a base for **JUMBO**, the Java library for CML, and other software described during this work.

## 1.5 Open Source software

Open Source software (OSS) is software for which the human-readable source code is made available under a copyright license that meets the Open Source definition[38]. The availability of OSS reduces the duplication of effort and allows developers to build on the work of others.

There are many examples of OSS in cheminformatics, a number of which are used in this thesis, such as the **CDK** (Chemistry Development Kit)[39] (for 2D structure generation), **OpenBabel**[40] (for substructure search) and **Jmol**[41] (for 3D structure visualization). All of the software libraries used in this work are OSS, and in turn all of the software created is provided as OSS.

## 1.6 e-Science

If we have access to large amounts of Open, machine-understandable data, then a problem emerges as to how we can store, retrieve and process it with

low expenditure and in reasonable time frames. In order to engage the problem of managing and processing the large quantity of data being produced, the UK e-Science Programme began in 2001 as a five-year coordinated initiative involving all the Research Councils and the Department of Trade and Industry. The term e-Science has been described as:

...the large scale science that will increasingly be carried out through distributed global collaborations enabled by the Internet. Typically, a feature of such collaborative scientific enterprises is that they will require access to very large data collections, very large scale computing resources and high performance visualisation back to the individual user scientists.[31]

These large scale resources are provided by what have been termed as *grids*, so called because they consist of a computer cluster of multiple, possibly geographically dispersed, nodes. The power of grid computing is highlighted by the Folding@home project[32] which, on 2007-11-30, has over 264,000 active distributed processing units[33]. These have combined to give a performance level of over 1.3 petaFLOPS\*, more powerful than any supercomputer in the world[34].

As well as a National e-Science Centre[35], the UK e-Science programme consists of a wide range of resources, including various regional e-Science centres. We have access to the Cambridge e-Science Centre[36], which provides the computational grid, CamGrid[37], which is used heavily in chapters 5 and 6 of this thesis.

## 1.7 Aims

The goals of this work were to:

- help extend Tim Berners-Lee's vision for the Semantic Web to chemistry, where chemical information on the Web is made Open and given well-defined meaning,

---

\*Floating point Operations Per Second

- show how Open, semantic data assists e-Science.

The work can be split into three sections:

1. Chapters 2, 3 and 4 describe the work undertaken to create a visible, Web-based, Open collection of semantic crystallographic data.
2. Chapter 5 shows how this collection enabled the performance of high-throughput calculations to validate both the data and a semi-empirical method for calculating solid-state geometry.
3. Chapter 6 provides another example of using Open Data to validate the data and a computational NMR method. This work also highlights the collaborative effect of making data and analysis Openly available. By providing up-to-date webpages summarizing the work, interested users were able to take part in *crowdsourcing*, which helped in the discovery of several data errors.

## Chapter 2

# Locating chemical data on the Web

### 2.1 Introduction

As the amount of chemical information on the Web increases and the number of chemical sites grows, a single entry-point to search over all available data becomes a necessity. A possible solution for this is to use commercial search engines, which provide text-based searches over large portions of Web content. These have no native functionality to assist chemical search, and so chemical identifiers must be made available to their indexes and queried like any other web content, *i.e.* as text strings.

The work contained in this chapter examines the aspects of the ideal string representation of chemical structures, and discusses the different identifiers currently in use. Investigations are made into the ability of these identifiers to represent chemical structures uniquely, as well as providing details of their efficacy as Web queries. Also contained is the discussion of generic Web technologies which aid the use of commercial search engines for chemical search.

### 2.2 Web search-engines

Web search engines consist of three major elements:

1. A *spider* (also known as a crawler or robot). The spider works by visiting a webpage, reading it and then following links to the other pages within the site. A spider will return to a site on a regular basis to look for changes.
2. An *index*. A copy of every page the spider finds will be inserted into the index.
3. A *search interface*. This is the software that will search through the pages recorded in the index to find matches to a search, and rank them in order of what it believes is most relevant before returning the results to the querier.

The majority of today’s web searches are performed by a user supplying a series of keywords as a query on a search engine homepage. The engine will then examine its index for pages containing some or all of those keywords and will provided the user with a listing of best-matching pages according to its criteria (*e.g.* taking into account the occurrence and proximity of the keywords on those pages).

### 2.2.1 Sitemaps

If commercial search engines are to be relied on for universal chemical Web searches, it is necessary for their spiders to index all webpages that are chemically interesting. For sites containing very large collections of webpages (*e.g.* PubChem[15]), these spiders do not try to index the whole site in one go, as they have many millions of other sites to index. Instead they will index a small portion at a time, working their way down and along the page hierarchy, whilst also checking all previous pages that have been indexed. This can lead to long delays between a page being published and being indexed. Indeed, sites whose pages change often, or those that grow rapidly may never be indexed completely[42].

In order to counter this problem, the major search engines have united to support the Sitemaps Protocol[43], which is an XML format that allows

webmasters to inform spiders of vital information, such as which pages of a site are available for crawling and when they were last changed. It is clearly important for rapidly growing, large online chemical databases to adopt this if search engines are to provide exhaustive chemical Web searches.

## 2.3 Representing chemical entities as strings

As engines index the complete text of webpages, regardless of content, pages containing chemical terms can be retrieved by these textual searches. Thus, providing a search engine with the query *iodomethane* would theoretically return all webpages containing a reference to that compound.

In order to make chemical entities indexable and retrievable by Web search engines, we must represent them as strings in webpage text. In addition to this, it is essential to use string identifiers that,

- are unique, so multiple searches are not required to retrieve all instances,
- are available to anyone *i.e.* everyone should be able to derive the identifier for a particular entity,
- return few or no false positives (caused by the query string being found in a non-chemical context, or in another chemical name as a substring).

There are several ways of representing chemical entities as strings, here we discuss how well they match the criteria described above.

### Chemical formulae

Almost all molecules have isomers e.g.  $\text{C}_2\text{H}_6\text{O}$  could represent ethanol or dimethyl ether. Hence a web query using only the chemical formula therefore often retrieves many webpages for unwanted isomers.



## Systematic chemical names

An IUPAC (International Union of Pure and Applied Chemistry) name should, in principle, be unique, but it is rarely used outside patent and regulatory submissions. Thus, for a given compound there will be many different names on the Web e.g. ‘methanol’ can also be found as ‘methyl alcohol’ or ‘methyl hydroxide’. To find all information about a compound on the Web using this method would require a lookup and querying using all possible systematic chemical synonyms.

## Arbitrary chemical names

Many compounds possess more than one arbitrary name. Methanol is also known as ‘wood alcohol’, ‘wood spirit’ and ‘colonial spirit’[44]. As with systematic chemical names a lookup is required to ascertain all arbitrary names of a compound.

## Semantically-free identifiers

Examples of these identifiers are any registry number systems such as CAS[45] or RTECS[46]. While it is possible to uniquely identify all chemical structures with such identifiers, they bear no relation to the structure they represent. Such systems rely on a lookup of the registry number, which first requires knowledge of a matching systematic or arbitrary chemical name. I have shown that such identifiers are liable to retrieve many false positives when used as web based queries (appendix A.3).

## Linear connection tables

Some covalent molecules can be well defined by a labelled graph of the atoms (nodes) and bonds (edges) between them. For example propane will always be written CH<sub>3</sub>-CH<sub>2</sub>-CH<sub>3</sub>. For such molecules the bond positions, bond orders, hydrogen atoms and formal charges are absolute. For other molecules the ordering of atoms can differ *e.g.* propan-1-al could be represented by CH(=O)-CH<sub>2</sub>-CH<sub>3</sub> or CH<sub>3</sub>-CH<sub>2</sub>-C(=O)H. Other problems include

- arbitrary multiple bond position, as in resonance structures of aromatic systems,
- mobile hydrogen atoms (tautomerism),
- formal charges that can be redistributed.

Therefore, for most molecules this type of connection table is not a unique identifier.

### Canonical identifiers

These identifiers aim to address the problems of connection table (CT) representation by converting the chemical structure (in the form of its CT) to a semantically-rich, unique canonical serialization of characters by fixed algorithms. Two requirements must be fulfilled in doing this:

- Different compounds must have different identifiers, with all the information needed to distinguish the structures.
- Any one compound has only one identifier, including only the necessary information to identify that compound.

An example of such an identifier is SMILES[47][48] (Simplified Molecular Input Line Entry Specification) notation. SMILES does not solve all problematic aspects of CT representation (e.g. mixtures of stereoisomers), but does provide a method for canonicalizing the ordering of heavy atoms and representation of stereochemistry. For instance *L*-glutamic acid is denoted as

C(CC(=O)O)[C@@H](C(=O)O)N

where the @ symbols denote the stereochemistry of the chiral centre. SMILES is widely used on websites containing chemical information and in online chemical databases, and as I have shown has high precision when used as a web based query (appendix A.5). However, as SMILES is proprietary and is a Closed project, different implementations of the generation algorithm are

in use, leading to different SMILES versions of the same compound. In fact, I found seven different SMILES representations for caffeine on webpages\*:

- [c]1([n+]( [CH3] ) [c] ([c]2([c] ([n+]1[CH3]) [n] [cH] [n+]2[CH3])) [O-]) [O-]
- CN1C(=O)N(C)C(=O)C(N(C)C=N2)=C12
- Cn1cnc2n(C)c(=O)n(C)c(=O)c12
- Cn1cnc2c1c(=O)n(C)c(=O)n2C
- N1(C)C(=O)N(C)C2=C(C1=O)N(C)C=N2
- O=C1C2=C(N=CN2C)N(C(=O)N1C)C
- CN1C=NC2=C1C(=O)N(C)C(=O)N2C

So, to perform a global Web search for a chemical structure using SMILES notation you would need access to all implementations of the generation algorithm.

Another example of such an identifier was officially released in April 2005, the IUPAC International Chemical Identifier[50], which is described below.

## 2.4 The IUPAC International Chemical Identifier

The aim of the IUPAC International Chemical Identifier (InChI) project is to:

...establish a unique label, the [InChI], which would be a non-proprietary identifier for chemical substances that could be used in printed and electronic data sources thus enabling easier linking of diverse data compilations.[51]

---

\*Throughout this chapter, examples are used from the Unofficial InChI FAQ[49], which I have written.

The current InChI version expresses chemical structures in a standard machine-readable (ASCII) format, in terms of atomic connectivity, tautomeric state, isotopic enrichment, stereochemistry and electronic charge. It can represent neutral and ionic well-defined covalently bonded organic molecules, and also, with straightforward extension, inorganic, organometallic and coordination compounds.

### 2.4.1 InChI creation and structure

An InChI identifier is created from an input CT (in MOL, SDF or CML format) in three steps

- Normalization — conventions are removed while maintaining a complete description of the compound. Steps involved are
  - Ignore electron density and use simple atom connectivity only.
  - Disconnect salts and metal atoms in organometallic compounds.
  - Normalise mobile-hydrogens, variable protonation and charge.
- Canonicalization — a set of atom labels are algorithmically generated that do not depend on how the structure was initially drawn.
- Serialization — the set of labels derived during canonicalization are converted into a string of characters, the InChI.

The ability of InChI to provide one identifier for different representations of the same compound is highlighted in figure 2.1.

The serialized InChI string is composed of layers and sub-layers (but no sub-sub-layers – figure 2.2). Each layer holds a distinct and separable class of structural information, with the layers ordered to provide successive structural refinement. All layers and sub-layers start with `/?` (except for the chemical formula sub-layer of the Main Layer which starts with `/'`) where `?` is a lower-case letter to indicate the type of information held in that layer. For instance, atom connection starts with `/c` and the hydrogen/mobile-hydrogen sub-layer starts with `/h`.

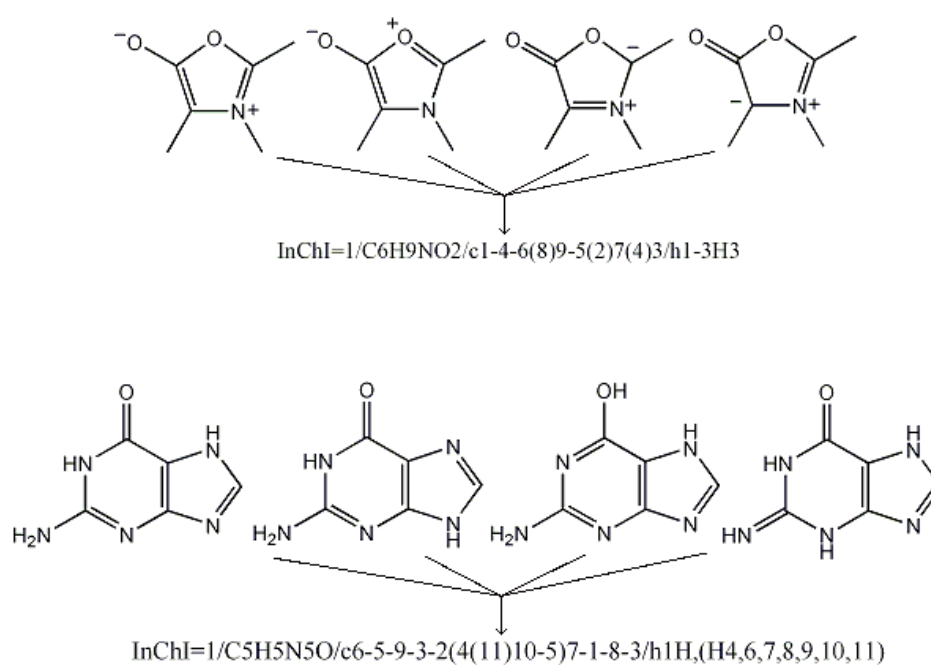


Figure 2.1: Different representations of the same compound have the same InChI, such as for the different resonance forms of the munchnones (top) and the tautomers of guanine (bottom).

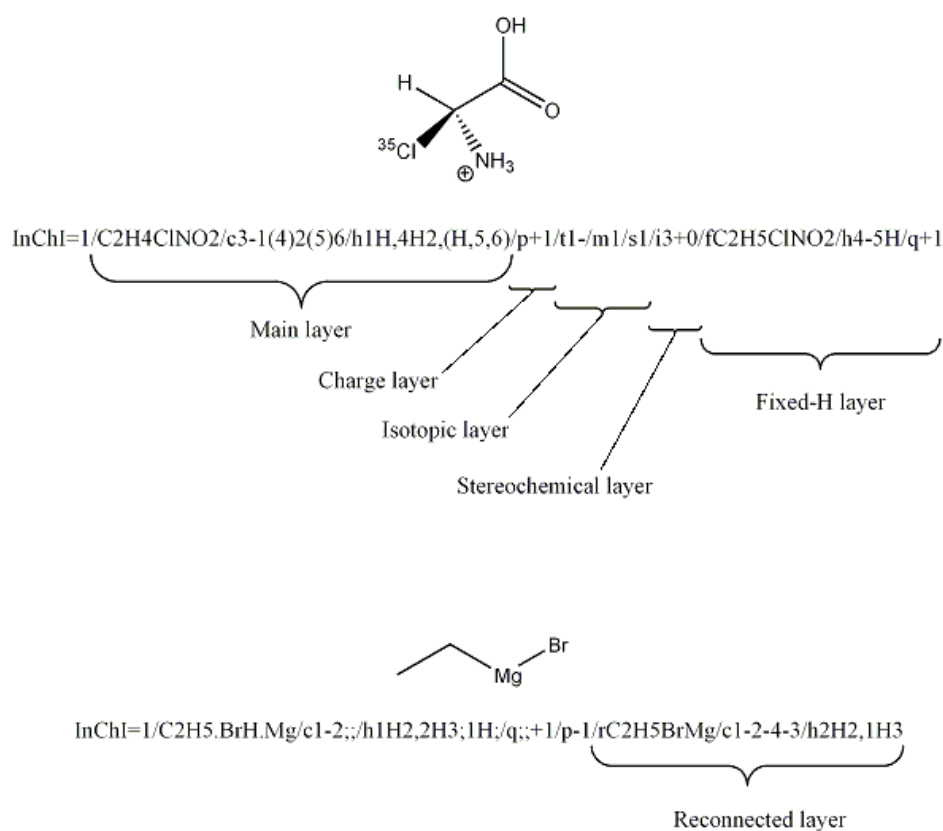


Figure 2.2: The classes of structural information that can currently be represented in an InChI

- Main layer - holding chemical formula, heavy atom connection and hydrogen/mobile-hydrogen connection sub-layers
- Charge layer - holding protonation level and charge sub-layers
- Stereochemical layer - holding sp<sup>2</sup> and sp<sup>3</sup> stereochemistry sub-layers (which include whether chirality is relative, absolute or racemic)
- Isotopic layer - the position and isotopic number of any atoms of a specific isotope.
- Fixed-H layer - an optional layer appended to the normalized InChI of a structure containing mobile-hydrogens to represent a particular tautomer.
- Reconnected layer - an optional layer appended to the normalized InChI of an organometallic compound to represent the structure using the conventions in the input CT.

The layered structure of the InChI allows future extensions for the representation of new classes of structural information with no change to the layers currently used. Work by the InChI task group has already begun on the inclusion of layers to represent polymeric structures and Markush structures.

The layered model allows chemists to represent chemical substances at a level of detail of their choosing. Except for the Main layer, the presence of a layer is not required and appears only when corresponding input information has been provided (figure 2.3).

As the InChI generation software is Openly available, anyone with an Internet connection can download it and generate InChI identifiers, or perhaps use it as an extension to their software (see section 2.4.3). This means that there will only ever be one implementation of the algorithm, and hence only one identifier per compound (see figure 2.4, as opposed to the earlier SMILES caffeine example).

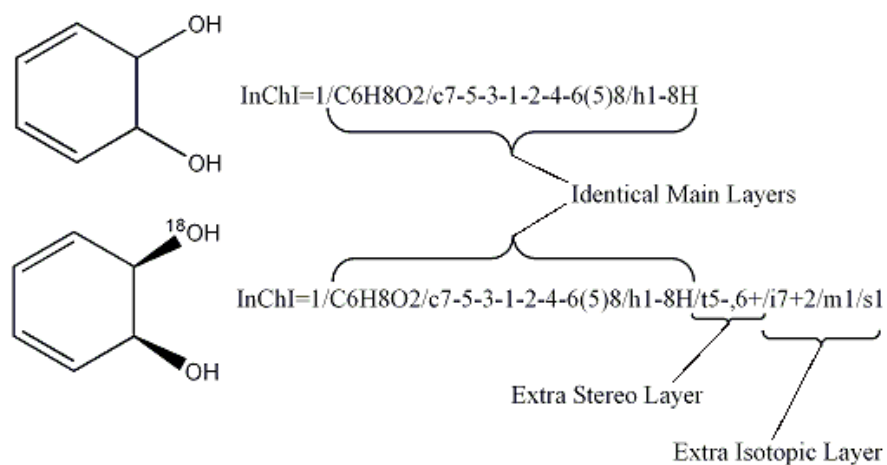
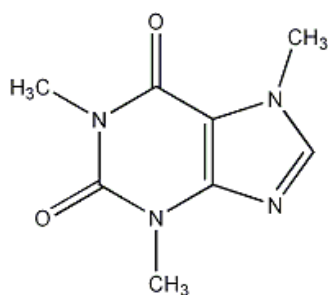


Figure 2.3: The InChI layer structure allows chemical entities to be represented to the known level of detail



InChI=1/C8H10N4O2/c1-10-4-9-6-5(10)7(13)12(3)8(14)11(6)2/h4H,1-3H3

Figure 2.4: The InChI representation of caffeine.



## 2.4.2 Searching for InChIs on the Web

Using InChIs provided for structures in the eCrystals archive[52] at the University of Southampton, I have shown that Web search engines can index and retrieve InChIs contained in webpages with almost 100% recall and 100% precision (appendix A.4). This can be attributed to the unusual form of an InChI, which includes a mix of letters, digits and generic punctuation.

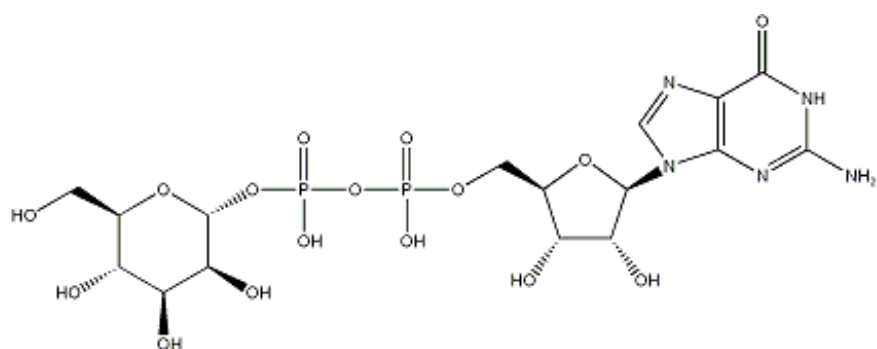
It was also shown (appendix A.6) that the length of InChIs is a problem when used as a query in current Web search engines. Many InChI strings contain more tokens than the engines will currently search for e.g. Google limits queries to 32 tokens and AltaVista to 20. Thus when searching for the InChI of a large molecule, it is possible for structural isomers to be returned (unrelated compounds will not match, as the empirical formula is one of the first tokens in an InChI string). This in effect reduces the uniqueness of the InChI, and means that some post-processing would need to be performed on the returned pages to find only the correct matches. Figure 2.5 highlights this by showing that, as Google would not include the stereochemical layer when searching for GDP-*D*-mannose, that its InChI would be seen as identical to that of GDP-*L*-mannose.

### InChIKey

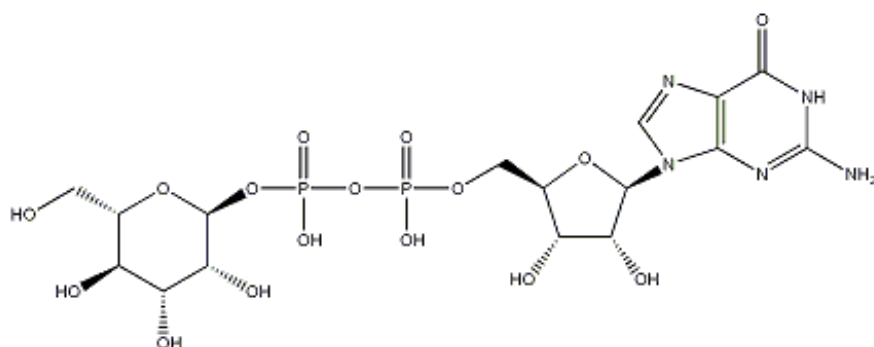
As a result of InChIs being shown to be generally too long to use as exact queries with Web search engines, the InChIKey was developed by the InChI team[53]. This is a 25 character hash code of the InChI string based on a truncated SHA-256 cryptographic hash function. It is made up of 4 parts

AAAAAAAAAAAAAA-BBBBBBBBCD

- (A) 14 characters for the basic structure
- (B) 8 characters for the layers
- (C) 1 character is a ‘check’ character



InChI=1/C16H25N5O16P2/c17-16-19-12-6(13(28)20-16)18-3-21(12)14-10(26)8(24)5(34-14)2-33-38(29,30)37-39(31,32)36-15-11(27)9(25)7(23)4(1-22)35-15/h3-5,7-11,14-15,22-27H,1-2H2,(H,29,30)(H,31,32)(H3,17,19,20,28)/t4-,5-,7-,8-,9+,10-,11+,14-,15-/m1/s1



InChI=1/C16H25N5O16P2/c17-16-19-12-6(13(28)20-16)18-3-21(12)14-10(26)8(24)5(34-14)2-33-38(29,30)37-39(31,32)36-15-11(27)9(25)7(23)4(1-22)35-15/h3-5,7-11,14-15,22-27H,1-2H2,(H,29,30)(H,31,32)(H3,17,19,20,28)/t4-,5+,7-,8+,9+,10+,11+,14+,15-/m0/s1

Figure 2.5: GDP-*D*-mannose (top) and GDP-*L*-mannose (bottom) and their corresponding InChI representations. The bold text shows the parts of the InChIs after the 32nd token (as defined by Google) and hence those parts that would not be used if the string were to be entered as a Google query.

- (D) 1 character is a flag indicating certain features (*e.g.* fixed or not-fixed hydrogens)

For instance, the InChIKey representation of caffeine is

**InChIKey=RYYVLZVUVIJVGH-UHFFFAOYAW**

- First block (14 characters) encodes molecular skeleton – RYYVLZVUVIJVGH
- Second block (8 characters) encodes proton positions, stereochemistry, isotopes, reconnected layer – UHFFFAOY
- Flag character indicates InChI version, presence/absence of fixed-H layer, isotopes and stereochemistry – A
- Check character – W

While the form of the InChIKey is different to that of an InChI string, it is expected that its unusual form will lead to similar performance when used as a query in Web search engines (*i.e.* high recall and precision), though no formal investigations have yet been made into this.

The use of InChIKey rather than InChI removes the possibility of searching for structures at differing levels of detail, *e.g.* by removing the stereochemical layer of the GDP-mannoses (figure 2.5), you could use the remaining string to do a search for either (though as discussed, this is not currently possible for long InChIs due to search engine limitations). The InChIKey aids exact structure searching at present, though if the search engine token limits were removed, it would be preferable to revert to using InChI for chemical search.

### **InChI/InChIKey adoption**

The fact that the InChI generation software is Openly available has led to widespread adoption by the chemical community. Rather than adopt one or the other, members of the chemical community generally provide both for their structures. Examples of databases which use InChI/InChIKey are[54]:

- PubChem – 11 million structures
- NCI DTP – 30 million structures
- ChemSpider – 17 million structures
- CrossFire Beilstein Database – 10 million structures
- ChemSure Patents – 10 million structures

InChI/InChIKey has also been adopted by many bloggers, who tag each entry with identifiers for the structures discussed within, *e.g.* Chem-bla-ics[55], Totally Synthetic[56] and Useful Chemistry[57]. The RSC’s Project Prospect[24] has shown how InChI/InChIKey can be embedded into article full-text and RSS feeds.

The InChI generation software has now been adopted into chemical drawing packages such as ChemDraw, ChemSketch and Marvin. Providing this capability directly into the chemist’s workflow is an important step, as it requires little effort to generate an InChI for each structure drawn.

### 2.4.3 The Google-InChI Web Service

In early 2005, Yong Zhang and I created the Google-InChI Web Service, which uses InChI to provide a method of performing chemical Web searches by drawing the structure you wish to find.

To do this, we provided a webpage that incorporates a Marvin[58] applet, into which a chemical structure can be drawn (figure 2.6). On clicking the ‘Search’ button, the following steps are performed

1. The drawn structure is output as a MOL file by Marvin.<sup>†</sup>
2. The MOL file is passed to the InChI executable and converted into an InChI string.

---

<sup>†</sup>This work was undertaken before the incorporation of the InChI software in ChemAxon products.

3. The InChI is submitted as a query to Google using the Google SOAP Search API[59].
4. The results are retrieved from Google as a SOAP response[60] and converted into a webpage.
5. The results webpage is displayed in the users browser (figure 2.7).

The example in figures 2.6 and 2.7 shows 51 different webpages that contain the InChI for naphthalene are retrieved. This shows how ‘Googling’ for InChIs in this way could provide a robust method for searching for chemistry on the Web in the future.

The Google SOAP Search API has now been deprecated, however, it is still possible to programmatically submit InChI queries to Google by concatenating the InChI string to:

`http://www.google.com/search?q=`

and submitting it as a URL. For instance, the Google query URL for naphthalene is:

`http://www.google.com/search?q=%22InChI%3D1%2FC10H8%2Fc1-2-6-10-8-4-3-7-9%2810%295-1%2Fh1-8H%22`

which will return the same page as if you were to go to the Google homepage and enter the naphthalene InChI string as a query. This technique can be used for all search engines, and has subsequently been implemented at the InChImatic website[61], which provides the same kind of 2D structure search as the Google-InChI service.

#### 2.4.4 Staurosporine - an InChI case study

Staurosporine is a natural product originally isolated in 1977 from the bacterium *Streptomyces staurosporeus* by Omura *et al*[62]. Later it was found to show strong inhibitory activity against protein kinases. The chemical structure of Staurosporine was elucidated by X-ray analysis of a single crystal[63]

WWMM Web Services - Mozilla Firefox

File Edit View Go Bookmarks Tools Help del.jcio.us

http://wwmm.ch.cam.ac.uk/wwmm/html/googleinchiserver.html

# World Wide Molecular Matrix

Home InChI CMLRSS OpenBabel CIF2CML JUMBO Tools

Generate InChI Google-InChI

## Search the Web Using InChI™ and Google™ ( times)

This Service searches the public Web for molecules whose InChIs have been indexed by Google. See [the InChI FAQ](#)

InChI version: 1

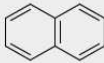
File Edit View Insert Tools Help

H C N O F React Select Erase Paste Undo Redo Zoom

- + P S Cl →

More Br I

□ □ □ □ □ □ □ □



Search

Applet MSketch started

- Create a molecule in the drawing space of the applet. (If you want a test, select naphthalene (the 2-ring molecule) and drag it to the drawing area)
- click "Search" ("Lucky search" gives the first hit only)
- You should get a list of links to pages containing that molecular structure
- If you get no hits, that is because no web page containing that InChI has been indexed yet - see FAQ. Check that naphthalene works (normally 3 or more hits).

Figure 2.6: The Google-InChI webpage with naphthalene drawn in the structure editor.



Figure 2.7: The results page for the Google-InChI service after naphthalene was submitted.

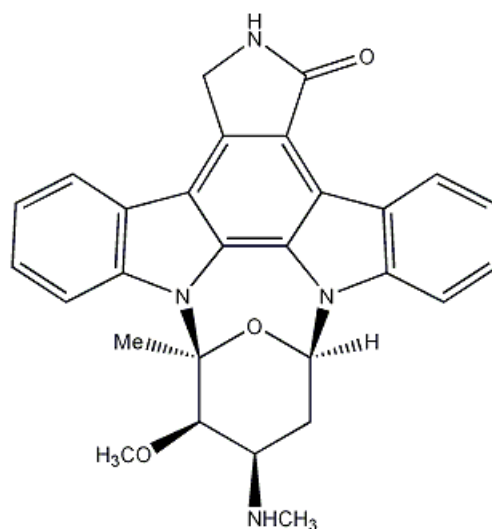


Figure 2.8: Structure of Staurosporine showing the absolute stereochemical configuration of the 4 chiral centres.

and the absolute stereochemical configuration by the same method in 1994[64] (figure 2.8).

In the crystal structure, the pyran ring adopts the boat conformation. The indole carbazole group is planar and sits either completely above or below the plane of the tetrahydropyran ring (figure 2.9).

Staurosporine has 4 chiral centres, the atom arrangement around each of which is critical to the activity of the compound. Thus, a correct representation of Staurosporine must not only have identical atoms and atom connections, but also identical descriptions of the 4 chiral centres.

However, during a search of the Web I investigated 19 different sites containing 24 examples of Staurosporine. Of these 24 examples, only 5 had the correct complete representation of Staurosporine[65][66][67][68][69]. The other 19 incorrect representations can be divided into the following categories

- No stereochemical information provided — 3 examples (figure 2.10)



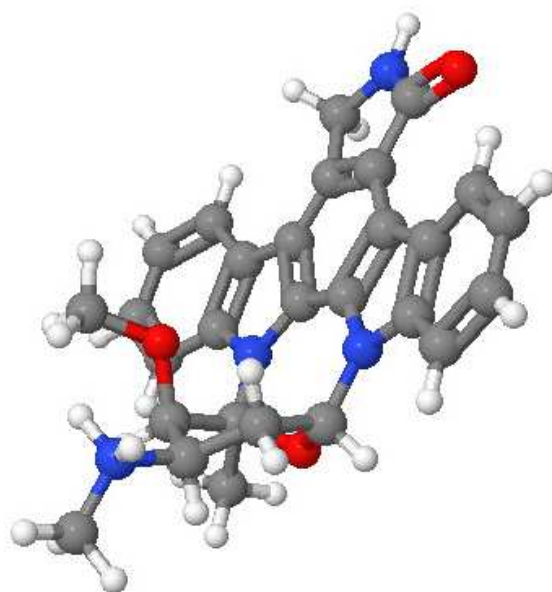


Figure 2.9: Conformation of Staurosporine in the solid state

- Partial incorrect stereochemical information provided — 2 examples (figure 2.11)
- All chiral centres defined, but incorrectly — 10 examples (figure 2.12)
- Incorrect atom connectivity — 2 examples (figure 2.13)
- Impossible stereochemistry — 2 examples (figure 2.14)

Interestingly, it seems as though the incorrect representation of Staurosporine has not propagated from one site through to others. In fact, every one has been drawn differently.

The InChI generated for the correct representation of Staurosporine is as below (see also figure 2.15).

```
InChI=1/C28H26N4O3/c1-28-26(34-3)17(29-2)12-20(35-28)31-18-
10-6-4-8-14(18)22-23-16(13-30-27(23)33)21-15-9-5-7-11-19(15)
```

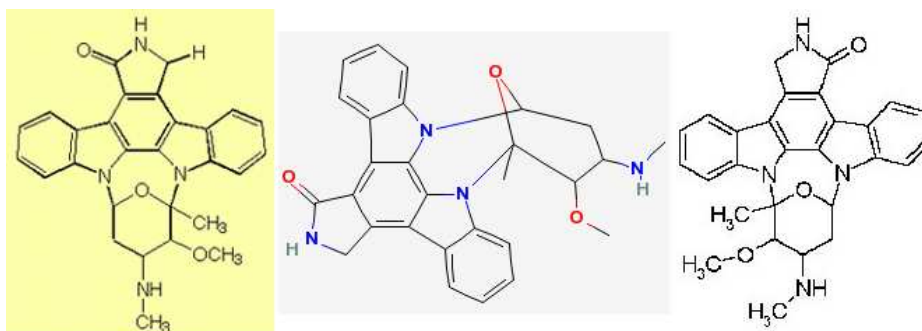


Figure 2.10: Representations of Staurosporine found on the Web with no stereochemical information provided[70][71][72]

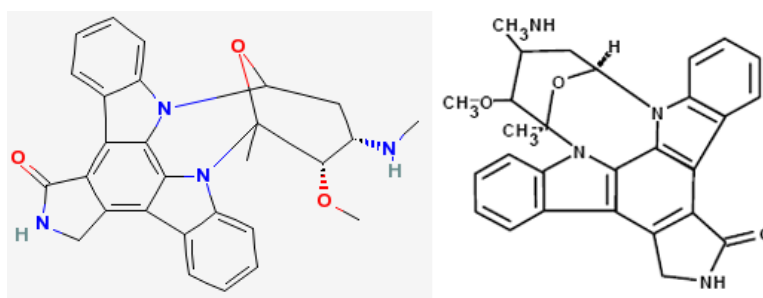


Figure 2.11: Representations of Staurosporine found on the Web with some stereocentres defined, but incorrectly[73][74]

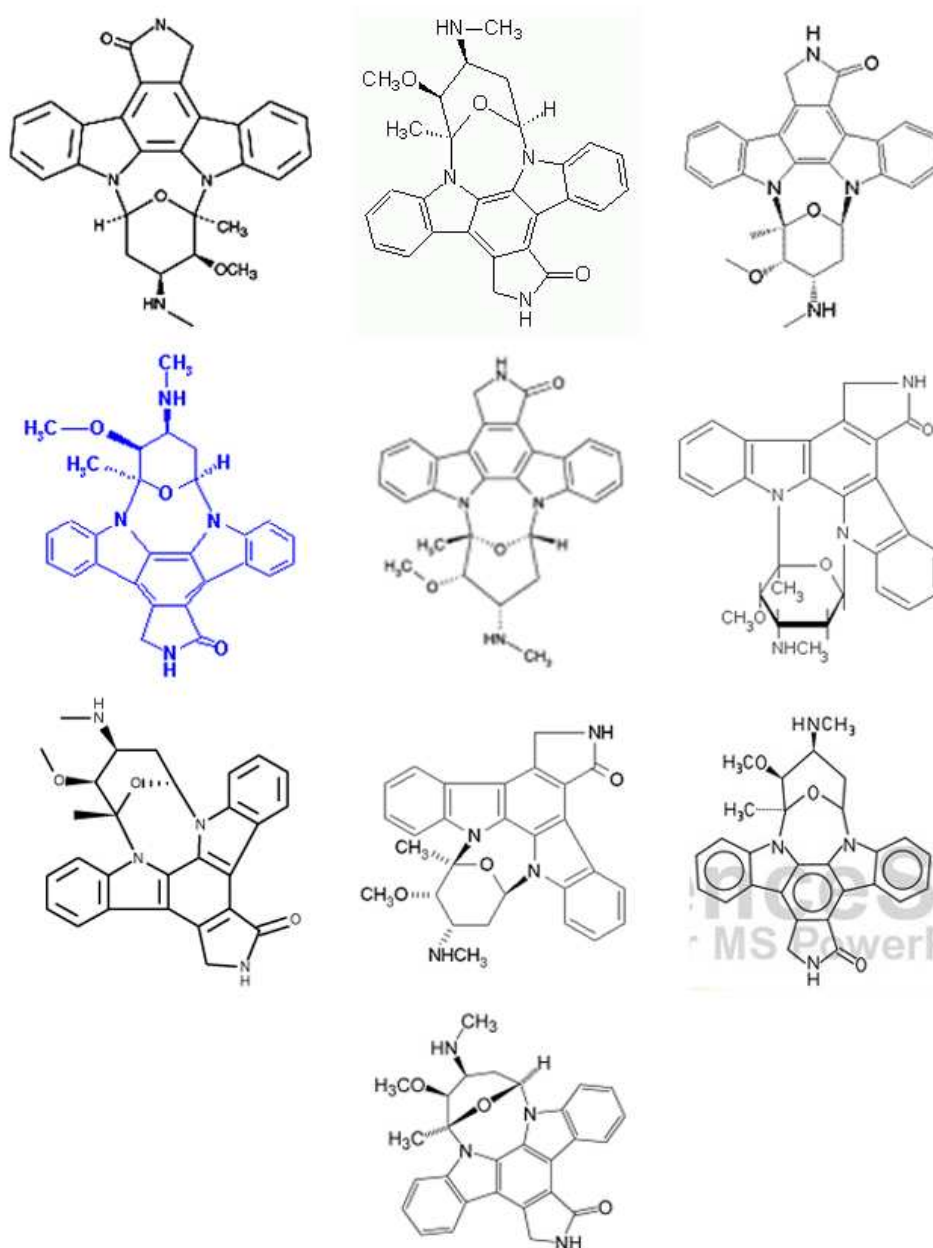


Figure 2.12: Representations of Staurosporine found on the Web with all stereocentres defined, but incorrectly (from left to right, top to bottom[75][76][77][78][79][80][81][82][83][84])

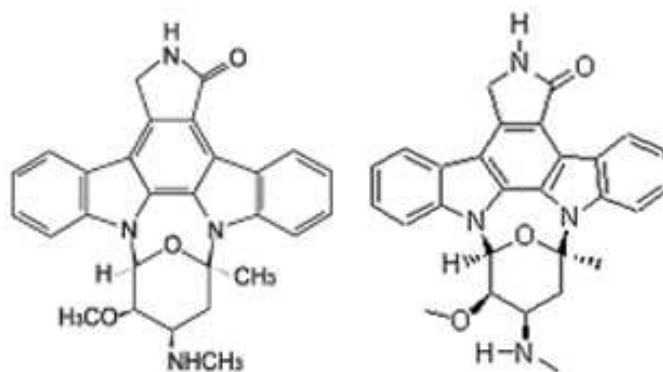


Figure 2.13: Representations of Staurosporine found on the Web with incorrect connectivity[85][86]

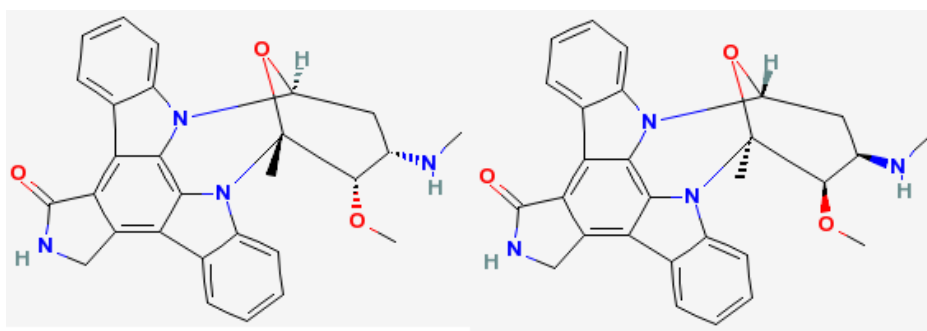


Figure 2.14: Representations of Staurosporine found on the Web with impossible stereochemistry[87][88]

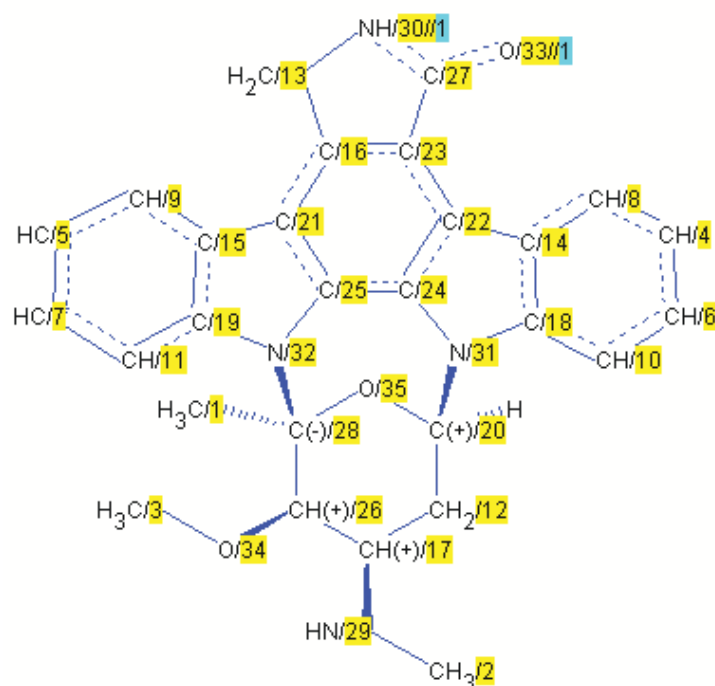


Figure 2.15: Staurosporine as it appears in the wInChI program. The numbers beside each atom represent the atomic canonical numbering. If there are two numbers beside an atom, the right-hand one denotes that a mobile-H is shared between atoms of the same number (i.e. 30 and 33)

32(28)25(21)24(22)31/h4-11,17,20,26,29H,12-13H2,1-3H3,(H,  
30,33)/t17-,20-,26-,28+/m1/s1

Note that numbers in an InChI string refer to the canonical numbering of all the atoms except non-bridging H. When broken up into its component layers this is

- InChI version layer: InChI=1
- Main layer, consisting of sub-layers
  - Chemical Formula: C28H26N4O3
  - Atom Connection: c1-28-26(34-3)17(29-2)12-20(35-28)31-18-10-6-4-8-14(18)22-23-16(13-30-27(23)33)21-15-9-5-7-11-19(15)32(28)25(21)24(22)31
  - Hydrogen/mobile hydrogen layer: h4-11,17,20,26,29H,12-13H2,1-3H3,(H,30,33) note that the mobile hydrogens are denoted in parentheses at the end of the sub-layer.
- sp3 stereochemistry layer: t17-,20-,26-,28+/m1/s1 note that compounds with identical stereochemical layers that differ only in containing m1 or m0 are enantiomers.

### InChIs for the incorrect representations of Staurosporine

The InChI below represents those structures with no stereochemical information given. All three of these structures give identical InChIs, which are the same as the InChI for Staurosporine minus the stereochemical layer.

InChI=1/C28H26N4O3/c1-28-26(34-3)17(29-2)12-20(35-28)31-18-10-6-4-8-14(18)22-23-16(13-30-27(23)33)21-15-9-5-7-11-19(15)32(28)25(21)24(22)31/h4-11,17,20,26,29H,12-13H2,1-3H3,(H,30,33)

The two InChIs below represent those structures with partial incorrect stereochemical information. As the previous structures, these have exactly the same InChI as the correct representation except for the stereochemical layer. In each of the structures the InChI generation program has noted that there are two chiral centres for which no stereochemical information is given and thus represents them with a '?' in the InChI.

```
InChI=1/C28H26N4O3/c1-28-26(34-3)17(29-2)12-20(35-28)31-18-
10-6-4-8-14(18)22-23-16(13-30-27(23)33)21-15-9-5-7-11-19(15)
32(28)25(21)24(22)31/h4-11,17,20,26,29H,12-13H2,1-3H3,(H,
30,33)/t17?,20-,26?,28+/m0/s1
```

```
InChI=1/C28H26N4O3/c1-28-26(34-3)17(29-2)12-20(35-28)31-18-
10-6-4-8-14(18)22-23-16(13-30-27(23)33)21-15-9-5-7-11-19(15)
32(28)25(21)24(22)31/h4-11,17,20,26,29H,12-13H2,1-3H3,(H,
30,33)/t17-,20?,26-,28?/m0/s1
```

The InChIs for structures with incorrect stereochemistry again differ from that of the correct representation only in the stereochemical layer. For these structures, all chiral centres have been defined, but they have different labels to the correct chirality. For example, the InChI for the structure in the top-left structure in figure 2.12 is

```
InChI=1/C28H26N4O3/c1-28-26(34-3)17(29-2)12-20(35-28)31-18-
10-6-4-8-14(18)22-23-16(13-30-27(23)33)21-15-9-5-7-11-19(15)
32(28)25(21)24(22)31/h4-11,17,20,26,29H,12-13H2,1-3H3,(H,
30,33)/t17-,20+,26-,28-/m0/s1
```

The two structures with incorrect connectivity are identical, with the Me group and H in the alpha position to the oxygen in the pyran ring being in each others respective correct position. Thus the InChIs for these two structures differs from the correct representation in atom connection sub-layer of the main layer.

```
InChI=1/C28H26N4O3/c1-28-12-17(29-2)25(34-3)27(35-28)31-18-
10-6-4-8-14(18)20-16-13-30-26(33)22(16)21-15-9-5-7-11-19(15)
32(28)24(21)23(20)31/h4-11,17,25,27,29H,12-13H2,1-3H3,(H,
30,33)/t17-,25-,27+,28-/m1/s1
```

The structures in figure 2.14 have impossible stereochemistry as the planar group attached to the pyran ring would have to be both above and below the ring to have such a structure. Interestingly, both these structures were found on the PubChem website which also provides the InChIs (as well as the SDF file from which the InChI is created). The InChIs given for the structures are

```
InChI=1/C28H26N4O3/c1-28-26(34-3)17(29-2)12-20(35-28)31-18-
10-6-4-8-14(18)22-23-16(13-30-27(23)33)21-15-9-5-7-11-19(15)
32(28)25(21)24(22)31/h4-11,17,20,26,29H,12-13H2,1-3H3,(H,
30,33)/t17-,20-,26-,28+/m0/s1
```

```
InChI=1/C28H26N4O3/c1-28-26(34-3)17(29-2)12-20(35-28)31-18-
10-6-4-8-14(18)22-23-16(13-30-27(23)33)21-15-9-5-7-11-19(15)
32(28)25(21)24(22)31/h4-11,17,20,26,29H,12-13H2,1-3H3,(H,
30,33)/t17-,20-,26-,28+/m1/s1
```

These InChIs are consistent with the structures in the SDF files, and show that the latter structure should actually be the correct representation of Staurosporine with the former being its enantiomer. This discrepancy could be caused by bugs in the image generation software, or perhaps the images are generated from a connection table other than that given in the attached SDF file.

This demonstration shows ability of InChI to distinguish between different chemical structures and the same structure with differing levels of detail. Due to the kind of inaccuracies shown, it is critical for chemical data providers to use technologies like InChI to perform internal and external consistency checking. Current work at ChemSpider[89] is investigating the efficacy of



community annotation and validation of errors in online chemical collections, and we have begun investigations with CrystalEye (section 4.3.8).

## 2.5 Conclusions

The InChI team has provided an excellent method for the unique representation of chemical structures as strings, with an important decision being made in making the InChI software Openly available. Not only has this led to the rapid and widespread adoption of InChI among the online chemical community, but it ensures that there need be only one generation tool, which means there should be no future issues as to the ‘correct’ InChI representation of a structure.

The unusual form of the InChI makes it a good choice for high recall and precision *exact* searches for chemical structures on the Web. However, current search engine limitations, such as the number of query tokens used and keyword breaking on punctuation, mean that many InChIs are not suitable in this manner. The InChIKey solves these problems by providing a fixed length hash of the InChI string, and so makes it the best choice at present for chemical structure search using commercial search engines.

## Chapter 3

# Creating and deriving data in CML from CIF

### 3.1 Introduction

The CIF[21] (Crystallographic Information File) is a community standard for exchanging and publishing crystallographic data. CIFs are routinely provided as supplemental data to published articles describing crystal structures, and as such there is a large volume of Open crystallographic data published in this way, albeit in a diffuse manner.

CIF has a well-defined syntax and set of extensible, machine-accessible dictionaries to define the data terms within. CIF documents are able to describe:

- crystal structure and composition (*e.g.* cell parameters, symmetry elements and atom site descriptions),
- experimental metadata (*e.g.* information about the measurement strategy, reduction methodology and solution and refinement procedures),
- researcher metadata (*e.g.* researcher names and institutions, contact details).

A public robotic CIF referee named checkCIF (further discussed in section 3.3.4) is used to validate the quality and completeness of data given in a high

percentage of published CIFs. Thus CIFs are a source of high quality, Open, machine-understandable data, and so ideal documents to use for automated validation of both data and theory. In order to assist reuse of data from CIFs, two tasks must be performed:

1. convert the data in the CIF format to CML,
2. derive the connection tables of the chemical structures contained with each CIF (unfortunately this is not provided explicitly at present).

This chapter details the software created to convert data in the CIF format to CML. This conversion is a three-step process:

1. convert the data in the CIF to an XML format (which is named CIFXML),
2. convert the CIFXML data to CML,
3. derive the connection table and other data from this CML and merge the two to create an ‘enhanced’ CML document.

As the software is all Open Source, by splitting the process up, users can choose which parts they wish to adopt. For instance, if a user disagrees with the methods of deriving data from the CML and wishes to do it themselves, then they can just use the libraries for performing the first two steps. The three steps are discussed in separate sections later in this chapter.

## 3.2 Why convert CIF to CML?

The CIF standard was developed before XML became a mainstream technology, though it has since been stated by the IUCr that

If the [CIF] standard were being developed afresh nowadays, it would probably have an underlying XML representation.[90]

The CIF structure is almost isomorphous to XML (there is no equivalent to the `loop` construct in XML), and the relationship between CIF and the

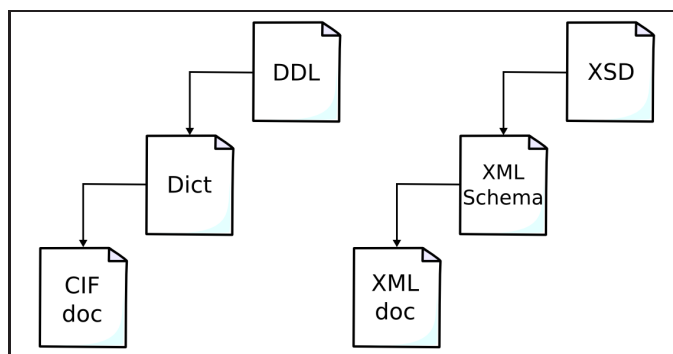


Figure 3.1: Diagram showing the similarity between the Crystallographic Information Framework and XML. The data items in a CIF document are defined in a CIF dictionary, the items of which are defined in a CIF Dictionary-Definition Language (DDL). Similarly, the data items in an XML document are defined in an XML Schema, which in turn is defined by the XML Schema Definition (XSD).

CIF dictionaries is analogous to that of XML and XML Schema/DTD (figure 3.1). A semantically lossless conversion of the CIF data into CML can be performed, which will subsequently give the benefits of being able to:

- use the large variety of tools available to process, manipulate, store, search and share XML documents. XML is the base language of the Semantic Web and there is a huge community continually creating and developing new generations of these tools.
- use chemically aware tools for CML (*e.g.* JUMBO, OpenBabel) to derive data,
- store data from more than one domain in the same file through the use of namespaces. Thus, we can store crystallographic data in CML along with MathML or include the CML in an RSS newsfeed.

### 3.3 The Crystallographic Information Framework

The Crystallographic Information Framework[91] provides a file structure and extensible domain ontology for crystallographic information. It consists of the Crystallographic Information File, whose syntax is a superset of the STAR file structure[92] and whose data items are defined by a set of CIF dictionaries, which are in turn described by dictionary definition languages (DDLs). It is important to note that the CIF has been designed to separate form (the syntax and document structure) from content (the terms, their meaning and the relationships between them).

The CIF was adopted in 1990 by the International Union of Crystallography (IUCr) to

...ensure that the information on crystal structures could be archived without introducing the keyboarding errors that are inevitable in a system requiring that tables of coordinates be re-typed at least three times, as was [previously the case].[93]

CIFs are now produced by crystallographic laboratory equipment and are the *de facto* method for publishing crystallography, where they are provided as supplemental data to articles describing crystal structures.

#### 3.3.1 STAR file concepts and syntax

The construction of a STAR file is based on the following:

- Each file contains a sequence of data blocks.
- Each data block contains a sequence of individual data items.
- There may be any number of data blocks and any number of data items within each data block.
- The data block represents a logical grouping of data, *e.g.* the data to describe a particular crystal structure.

- The identity of each data item within a data block is determined by a unique data name which precedes it in the file.
- Data items may be repeated in lists by placing them within a simple data loop structure.

The STAR file syntax and terminology is defined by six rules:

*text string* : a string of characters bounded by blanks, single quotes ('), double quotes ("), or by semi-colons (;) as the first character of a line. Any text string that contains spaces must be delimited with single or double quotes, otherwise spaces will be interpreted as delimiters. If the string extends over more than one line, it must be delimited by a semicolon appearing as the first character on the line.

*data name* : a text string starting with an underline (\_) character.

*data item* : a text string not starting with an underline, but preceded by a data name to identify it (a tag-value pair).

*data loop* : a list of data names, preceded by `loop_` and followed by a repeated list of data items.

*data block* : a collection of data names (looped or not) and data items that are preceded by a `data_` code record. A data name must be unique within a data block. A data block is terminated by another `data_` statement or the end of file.

*data file* : a collection of data blocks, the block codes must be unique within a data file.

No assumptions are made about the order of the data blocks or data items, other than the requirement that the character strings which identify data blocks, or data names within a block, must be unique. There are no restrictions regarding the placement of data names or data items within a data block, other than the requirement that the name must precede the item.

```

data_global
  _publ_contact_author_email      neperson@something.com
  _publ_contact_author_phone      '01234 567890'

loop_
  _publ_author_name
  _publ_author_address
  'N. E. Person'
;
Crystalville University
Anycity
Anyland 1234
;
  'A. N. Other'
;
Crystalville University
Anycity
Anyland 1234
;

```

Figure 3.2: An example STAR file data block showing typical tag-value data items and a loop construct

The STAR syntax makes no distinction between the type of a data item (*i.e.* number or string). The order and format of these strings in the file are irrelevant, except for the requirement that the data name precede the data item. Data on a line following a hash character ‘#’ is considered to be a comment, except if it is contained within a text string.

A data item, or a set of data items, may be repeated in a list. Such data items are preceded by a `loop_` keyword. Any data item, independent of its type, may be included in a loop. The only requirement is that the number of data items in a loop must be an exact multiple of the number of data names in the loop definition. Figure 3.2 shows an example data block with the various constructs that constitute the STAR syntax.

### 3.3.2 CIF syntax

The CIF syntax is defined by imposing restrictions on the STAR syntax. These are:

- Lines may not exceed 80 characters.

- Data names and block codes may not exceed 32 characters and are case insensitive.
- While data items in a STAR file may be of any type, the CIF Dictionary identifies whether a CIF data item is a number or a character.
- A data item is assumed to be a number if it starts with a digit, '+', '-' or a period '.' and is not bounded by matching single or double quotes or semicolons as the first character on a line.
- A number may be supplied as an integer, as a floating-point number, or in scientific notation. For example: 34.6, 3.45E1, 34.5(12), 3.45E1(12) are all versions of 34.5 with and without an estimated standard deviation (esd) of 1.2.
- 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.
- A data item is assumed to be of data type *character* if it is not a *number* or *text*.
- Only one level of `loop_` is permitted. Additional levels of repeated data must be stored as lists within a text field.
- Each CIF item has a default units code which is stated in the CIF Dictionary. If a data item is not stored in the default units, the units code is appended to the data name.

Data names defined for use in a CIF are separated into components to represent an internal hierarchy of data categories. The data names are of the form `_<category>_<topic>_<subtopic>`, though more than three levels of hierarchy may exist in a given category. For example, data referring to atoms in the crystal structure are in the `atom` category, which has two topics: one which describes atoms sites in the structure, `site`, and one which describes properties of the atom types that occupy these sites, `type`. These two



```

data_exptl_crystal_density_meas
  _name                '_exptl_crystal_density_meas'
  _category             exptl_crystal
  _type                numb
  _type_conditions      esd
  _list                both
  _list_reference       '_exptl_crystal_id'
  _enumeration_range    0.0:
  _units               Mgm^-3^
  _units_detail         'megagrams per cubic metre'
  _definition
;      Density values measured using standard chemical and physical
      methods. The units are megagrams per cubic metre (grams per
      cubic centimetre).
;

```

Figure 3.3: A data block from the coreCIF dictionary containing the definition of a measured crystal density data item

topics each have many subtopics, *e.g.* `_atom_site_type_symbol` identifies the chemical symbol of the atom occupying a particular site in the crystal.

### 3.3.3 CIF dictionaries

The set of data names, items and definitions that are fundamental to crystallography are described by the CIF Core Dictionary (coreCIF)[94]. This dictionary is intended for use in the description of small-molecule and inorganic structures. Dictionaries have also been provided to extend this core dictionary to describe more specific areas of crystallography, *e.g.* mmCIF for macromolecular structures[95]. A data name definition taken from coreCIF is shown in figure 3.3.

The formalisms employed to record CIF data names and their definitions and properties within the CIF dictionaries are described by two Dictionary Definition Languages, DDL1[96][97] and DDL2[98].

...[DDL1 is] used to define the dictionary of basic crystallographic items known as [coreCIF]. DDL2 was developed in order to provide the much tighter definitions needed for macromolecular crystallography where automatic computer handling of information is required...[99]

The relationship between CIF and CIF Dictionaries is analogous to the relationship between XML and XMLSchema/DTD.

The dictionaries and DDLs conform to the CIF syntax. This allows programs to be written to compare CIFs to their dictionaries and check whether the data names and value types used in the CIF were defined in the dictionary and whether the data values lay within the prescribed ranges, *e.g.* it could check that the density is given as a positive number (this has been implemented, as shown in section 3.5).

To maintain and develop the CIF standards, a committee (COMCIFs) has been appointed to ensure that all extensions to the CIF dictionary conform to the STAR syntax and do not violate any of the conventions of the current dictionary.

### 3.3.4 **checkCIF**

checkCIF[100] is a web service provided by the IUCr (see figure 3.4) which validates the data held in a CIF. The service subjects a CIF to a large collection of algorithmic tests (there are 346 currently listed[101]) which check:

- internal consistency of data dependencies,
- scientific plausibility of the model,
- completeness of experimental metadata,
- quality of the derived structural model.[90]

checkCIF is an integral part of the submission and review cycle for IUCr journals. Authors use it as a ‘robotic referee’ prior to formal submission to ensure that their files are free from syntax errors and gross crystallographic errors. Human referees use the warnings raised by checkCIF to assess the overall scientific argument presented in an article.

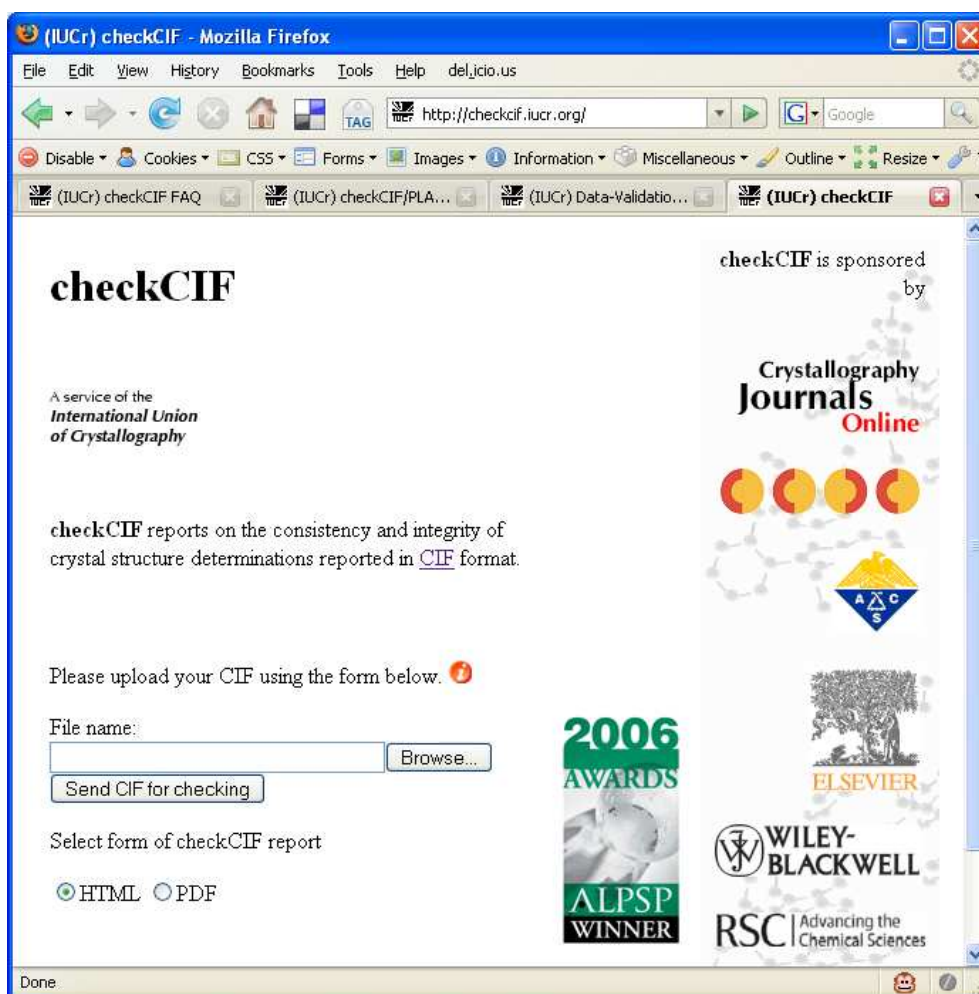


Figure 3.4: The checkCIF service homepage. The submission form is shown in the bottom-left.

The service is used through a webpage form, where a CIF can be uploaded from a remote computer and run through checkCIF at the click of a button. The checkCIF report is returned optionally as either a PDF or HTML page. Data that do not pass a test are reported as *alerts* of differing levels (A, B, C and G, from most to least severe). Authors are encouraged to fix as many problems highlighted by the alerts as possible. Any alert with level A must either be corrected, or the author must provide a valid explanation for its occurrence. Acta Crystallographica Section E now routinely provides the checkCIF report for each CIF published (an example is shown in figure 3.5).

Other publishers use checkCIF as part of the publishing process (as shown by the sponsors in figure 3.4), though it is not required for all journals. It has been noticeable during this work that the rate of syntactical and other errors in CIFs from the IUCr is far lower than that of other publishers.

### 3.4 CIFXML: converting CIF to XML

Programming libraries for working with CIFs have already been described for Fortran[102] and C, or variants[103][104], Python[105][106] and Perl[107]. We\* have created CIFXML, an XML dialect with a corresponding XML DTD and Schema. Alongside this, we have developed CIFXML-J, a Java library for converting CIFs to valid CIFXML and *vice versa*.

CIFXML-J allows CIFs to be read, edited, syntactically validated, sorted, normalised, filtered, stored as an XML DOM, transformed and output. It is based on the two main strategies for processing structured documents in XML, SAX[108] and DOM[109].

- SAX. After lexical processing, a document is broken into chunks, which fire events in a linear order. In XML this is normally for start and end tags and contained text.

---

\*The work on CIFXML was collaborative; Peter Murray-Rust, Simon Tyrrell and this author have all been involved; overall, this author contributed around 50% of the work. Where work was performed by a specific individual, this is indicated.

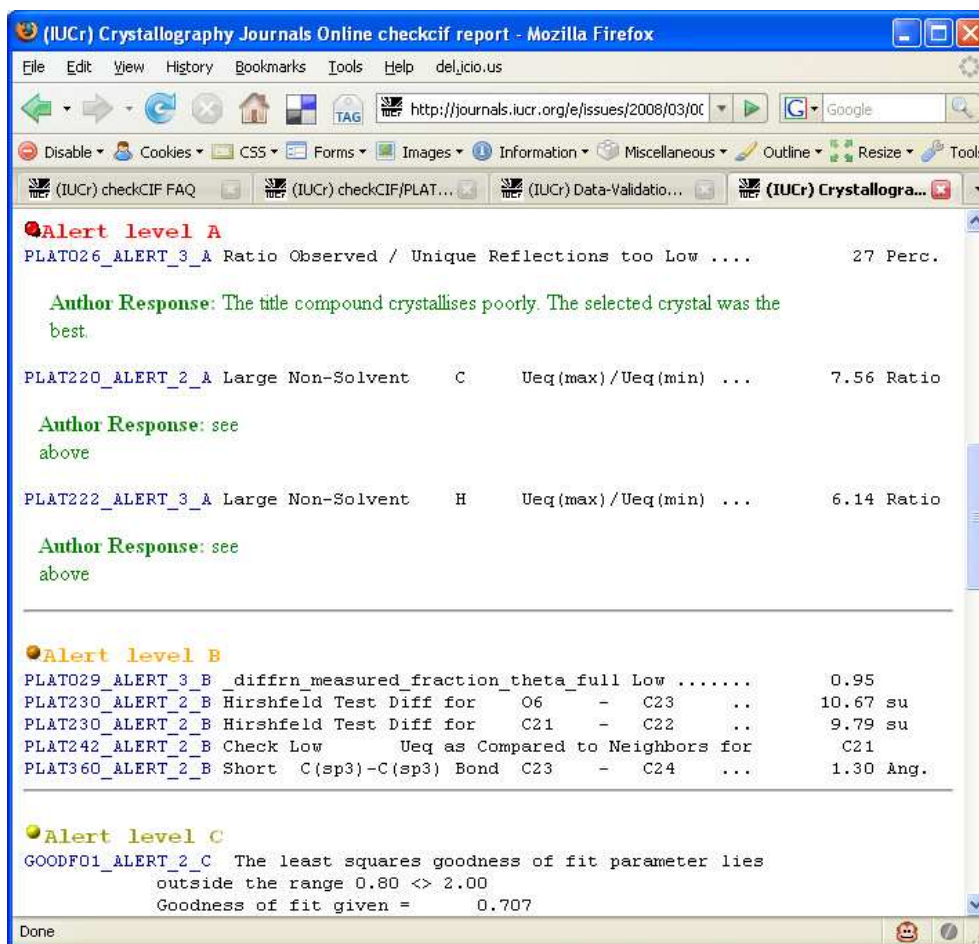


Figure 3.5: The alert section of a checkCIF report (in HTML) from Acta Crystallographica Section E. Note the author comments explaining the alerts with level ‘A’.

- **DOM.** The document is converted into a tree structure (often representable by a DTD or XML Schema). The tree is held in memory and can be navigated, and transformed in many ways.

SAX and DOM are complementary. SAX has the advantage of being rapid and not limited by memory. DOM preserves the context of every piece of information. In practice many XML parsers provide both strategies and use SAX to build a DOM. A brief description of the use of SAX, DOM and callbacks is given later.

The CIF standard requires that data instances are valid against one or more dictionaries. In practice few tools validate CIFs against any dictionary, though certain semantics can only be applied if a dictionary is available (*e.g.* the requirement that elements in a loop must belong to the same category). Dictionary validation is omitted from the core **CIFXML-J** engine. The dictionary validated transformation of CIFXML documents into CML will be described in section 3.5

CIFXML currently supports the CIF syntax and DDL1[97]-based dictionaries (but not STAR or DDL2[98]). It interprets any CIF as a structured document (**CIF**), which may contain

- **datablocks:** these must have unique ids and may contain **items**, **loops** and **comments**
- **items:** all item names must be unique within a **datablock**
- **loops:** all **loops** within a **datablock** must belong to different categories, and all names in the **loop** should be unique
- **comments:** **comments** can occur anywhere within a CIF as long as they do not break items or loops. It is unclear whether **comments** are technically part of the content of a CIF or simply annotations for human readers only. We deprecate their use for holding information but since they are often used for metadata we retain them in the CIFXML model

- Whitespace: CIF elements can be separated by inline and interline whitespace but this is not included in CIFXML data model.

The CIF syntax allows for a small number of syntactic variants such as delimiters on values or token used for whitespace. These are not held in the data model so will not be recovered in roundtrips.

As there is no formal concept of ordering in CIF, the data blocks, the elements within each and the components of a loop can be reordered without affecting the CIF. According to the specification the ordering of “rows” in a loop is not significant. However, XML supports the order of document elements and CIFXML preserves precisely all order in the input document. This allows CIFs to be “roundtripped” (*i.e.* read into the DOM and re-output without loss). In addition, the order of the components can be canonicalized so that it is possible to compare documents with arbitrary ordering but identical semantic content.

### 3.4.1 Conformance to CIF

CIFXML-J is used to establish the correctness of CIFXML to the XML Schema or DTD and to act as a CIF validator. CIFXML-J has been designed to implement the CIF standard as described in the specification. I have noticed, however, that a small but significant fraction of CIFs do not adhere to the specification precisely. The commonest deviations (which probably arise from using normal text-editors rather than CIF-aware ones) are

- Duplication of items
- Duplicate datablock names
- Incorrect use of delimiters
- Improper insertion of “comments” (sometimes apparently added by technical editors) which do not start with “#”
- Illegal characters (especially non-printing characters)

- The number of items in a loop not being exactly divisible by the number of data names provided

CIFXML-J provides optional heuristics to attempt recovery from these (see the example in section 3.4.5), but cannot, of course, guarantee that the result is what was intended. The first two points above can always be recovered from by taking the first item, or renaming the datablock respectively. The middle two points are dealt with only in certain cases, such as comments that appear before the start of the first datablock, or simple errors in the use of delimiters, *e.g.* it is common in early CIFs to see single quote delimited items spill over into a second line (which should then be delimited with ;). Comments appearing within loops, or missing delimiters are very difficult to deal with confidently, and so CIFXML-J will throw an error on encountering them. The latter two points should not be attempted to be fixed programmatically, and thus always cause errors to be thrown by the parser.

### 3.4.2 CIFXML-J functionality

CIFXML-J supports the following operations:

- Complete syntactic validation of CIFs
- Dictionary-free semantic validation against the CIF standard
- Conversion of escaped characters to their UNICODE equivalents
- Reporting of errors and warnings with original line numbers. Further processing after warnings. Attempted recovery from errors
- Optional parsing of numbers with standard uncertainty fields (*e.g.* 123.45(6))
- Choice of DOM or SAX strategies and choice of parsers
- Creation of a CIFXML object from CIF or XML
- Normalization of document structure



- Canonicalization of document structure
- Optional sorting of part or whole document
- Differences between data models for two CIFs (*i.e.* independent of syntax and ordering)
- Output as XML, HTML or CIF for round-tripping

### 3.4.3 Representation of CIFs in XML

We have created an XML DTD to which the CIFXML serialization of CIFs must conform:

```
<!DOCTYPE cif [
<!ELEMENT cif (datablock | comment)*>
<!ELEMENT datablock (comment | item | loop)*>
<!ATTLIST datablock
    id CDATA #REQUIRED>
<!ELEMENT comment (#PCDATA)>
<!ELEMENT item (#PCDATA)>
<!ATTLIST item
    name CDATA #REQUIRED
    su CDATA #IMPLIED
    numericValue CDATA #IMPLIED
    dataType CDATA #IMPLIED>
<!ELEMENT loop (row+)>
<!ATTLIST loop
    names CDATA #REQUIRED>
<!ELEMENT row (cell+)>
<!ELEMENT cell (#PCDATA)>
<!ATTLIST cell
    su CDATA #IMPLIED>
]>
```

which also closely reflects the data structure of the CIFXML. I have also created an XML Schema representation of this DTD:

```
<xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema'>
  <xs:element name='cell'>
    <xs:complexType mixed='true'>
      <xs:attribute name='su' />
    </xs:complexType>
  </xs:element>
  <xs:element name='cif'>
    <xs:complexType>
      <xs:choice minOccurs='0' maxOccurs='unbounded'>
        <xs:element ref='datablock' />
        <xs:element ref='comment' />
      </xs:choice>
    </xs:complexType>
  </xs:element>
```

```

<xs:element name='comment'>
  <xs:complexType mixed='true'>
    </xs:complexType>
  </xs:element>
<xs:element name='datablock'>
  <xs:complexType>
    <xs:choice minOccurs='0' maxOccurs='unbounded'>
      <xs:element ref='comment' />
      <xs:element ref='item' />
      <xs:element ref='loop' />
    </xs:choice>
    <xs:attribute name='id' use='required' />
  </xs:complexType>
</xs:element>
<xs:element name='item'>
  <xs:complexType mixed='true'>
    <xs:attribute name='name' use='required' />
    <xs:attribute name='su' />
    <xs:attribute name='numericValue' />
    <xs:attribute name='dataType' />
  </xs:complexType>
</xs:element>
<xs:element name='loop'>
  <xs:complexType>
    <xs:sequence>
      <xs:element ref='row' maxOccurs='unbounded' />
    </xs:sequence>
    <xs:attribute name='names' use='required' />
  </xs:complexType>
</xs:element>
<xs:element name='row'>
  <xs:complexType>
    <xs:sequence>
      <xs:element ref='cell' maxOccurs='unbounded' />
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>

```

A typical example of a CIF is:

```

data_global
loop_
  _publ_author_name
  _publ_author_address
  'A. Person'
;
University of Crystalville
Anycity
Anyland 1234
;
  'A. N. Other'
;
CrystalTown University
AnotherCity
Anotherland 5678
;
data_I

```

```

_audit_creation_method      SHELXL-97
_chemical_formula_sum       'C93 H84 Cl3 Co Fe N8 O2'
_chemical_formula_weight    1566.81

_symmetry_cell_setting      'Triclinic'
_symmetry_space_group_name_H-M 'P -1'

loop_
  _symmetry_equiv_pos_as_xyz
    'x, y, z'
    '-x, -y, -z'

_cell_length_a              13.8463(3)
_cell_length_b              16.8164(5)
_cell_length_c              17.9072(6)
_cell_angle_alpha           93.7800(10)
_cell_angle_beta            111.1430(10)
_cell_angle_gamma           97.4630(10)
_cell_volume                 3827.19(19)
_cell_formula_units_Z       2
_cell_measurement_temperature 110(2)

```

When expressed as CIFXML, this would be:

```

<cif>
  <datablock id="global">
    <loop names="_publ_author_name _publ_author_address">
      <row>
        <cell>
          A. Person
        </cell>
        <cell>
          University of Crystalville Anycity Anyland 1234
        </cell>
      </row>
      <row>
        <cell>
          A. N. Other
        </cell>
        <cell>
          CrystalTown University AnotherCity Anotherland 5678
        </cell>
      </row>
    </loop>
  </datablock>
  <datablock id="I">
    <item name="_audit_creation_method">SHELXL-97</item>
    <item name="_chemical_formula_sum">C93 H84 Cl3 Co Fe N8 O2</item>
    <item name="_chemical_formula_weight">1566.81</item>
    <item name="_symmetry_cell_setting">Triclinic</item>
    <item name="_symmetry_space_group_name_h-m">P -1</item>
    <loop names="_symmetry_equiv_pos_as_xyz">
      <row>
        <cell>x, y, z</cell>
      </row>
      <row>
        <cell>-x, -y, -z</cell>
      </row>
    </loop>
  </datablock>
</cif>

```

```

    </loop>
    <item name="_cell_length_a">13.8463(3)</item>
    <item name="_cell_length_b">16.8164(5)</item>
    <item name="_cell_length_c">17.9072(6)</item>
    <item name="_cell_angle_alpha">93.7800(10)</item>
    <item name="_cell_angle_beta">111.1430(10)</item>
    <item name="_cell_angle_gamma">97.4630(10)</item>
    <item name="_cell_volume">3827.19(19)</item>
    <item name="_cell_formula_units_z">2</item>
    <item name="_cell_measurement_temperature">110(2)</item>
  </datablock>
</cif>

```

An alternative syntax for the numeric fields is exemplified by:

```

<item name="_cell_length_a" su="0.0003"
  numericValue="13.8463" dataType="numb">13.8463(3)</item>
<item name="_cell_length_b" su="0.0005"
  numericValue="16.8164" dataType="numb">16.8164(5)</item>
<item name="_cell_length_c" su="0.0006"
  numericValue="17.9072" dataType="numb">17.9072(6)</item>
<item name="_cell_angle_alpha" su="0.0010"
  numericValue="93.7800" dataType="numb">93.7800(10)</item>
<item name="_cell_angle_beta" su="0.0010"
  numericValue="111.1430" dataType="numb">111.1430(10)</item>
<item name="_cell_angle_gamma" su="0.0010"
  numericValue="97.4630" dataType="numb">97.4630(10)</item>
<item name="_cell_volume" su="0.19"
  numericValue="3827.19" dataType="numb">3827.19(19)</item>
<item name="_cell_formula_units_z">2</item>
<item name="_cell_measurement_temperature" su="2.0"
  numericValue="110" dataType="numb">110(2)</item>

```

### 3.4.4 CIFXML-J architecture

CIFXML-J is a single package based closely on the SAX model. It contains the following main classes:

- AbstractBlock.java
- AbstractTextElement.java
- AbstractValueElement.java
- CIF.java
- CIFComment.java
- CIFContentHandler.java
- CIFDataBlock.java

- `CIFElement.java`
- `CIFErrorHandler.java`
- `CIFException.java`
- `CIFItem.java`
- `CIFLoop.java`
- `CIFParser.java`
- `CIFRow.java`
- `CIFSaveFrame.java`
- `CIFTableCell.java`
- `DOMBuilderContentHandler.java`
- `DefaultContentHandler.java`
- `DefaultErrorHandler.java`

The CIF parsing uses a SAX-like model where events cause callbacks to the Content- or ErrorHandlers. Classes that represent the basic elements of a CIF (as defined in the specification) have the element name prefixed by CIF, *e.g.* `CIFLoop` or `CIFDataBlock`. The inheritance hierarchy of the main CIFXML-J concrete classes is shown in figure 3.6. All CIFXML-J elements are descendants of the `XOM[30] Element` class.

The base class is `CIFElement` which defines a basic API for processes common to all subclasses.

- `String toCIFString ()`

This returns the `CIFElement` as a CIF-formatted String.

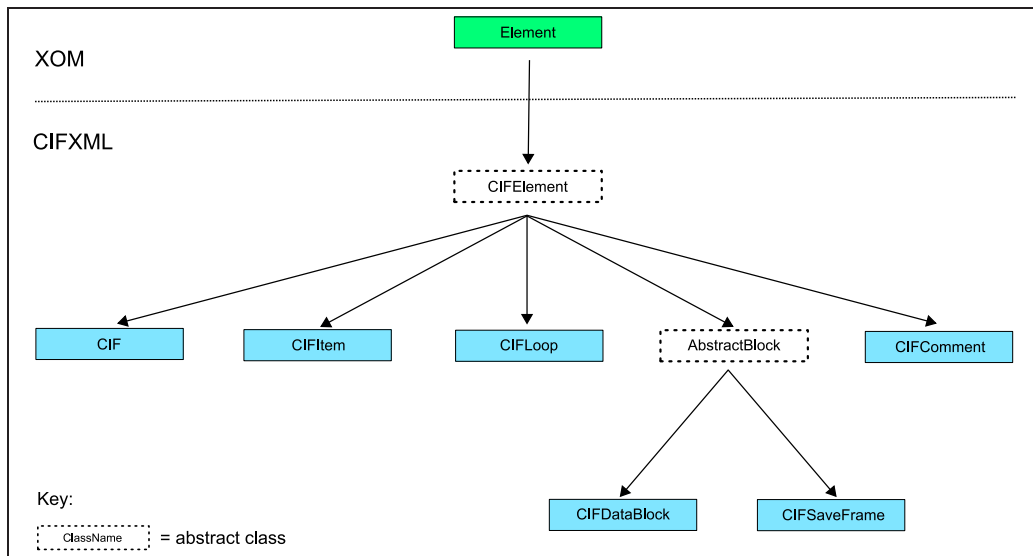


Figure 3.6: The CIFXML-J inheritance hierarchy (the CIFSaveFrame is reserved for expansion)

- `void writeXML (Writer w) throws IOException`

This will output the CIFELEMENT and all of its children in an XML format.

- `void writeHTML (Writer w) throws IOException`

This will output the CIFELEMENT and all of its children in an HTML format with lists converted into HTML tables.

- `void writeCIF (Writer w) throws IOException`

This will output the CIFELEMENT and all of its children in CIF format thus showing that CIFXML-J is a semantically lossless library. It uses the `toCIFString()` method described above.

- `void normalize ()`

This will attempt to remove any lexical variants.

- `void canonicalize ()`

Within a CIF the order of the datablocks, items and loops (including the row/column ordering) are all arbitrary. This will reorganise the order of the various `CIFElements` within a `CIFDocument` into a lexical order.

The default behaviour of `canonicalize()` is to apply the following heuristics during its reordering

- `CIFItems` occur lexically before `CIFLoops`.
- All `CIFItems` are sorted alphabetically by name.
- The columns of each `CIFLoop` are sorted alphabetically by namelist, then the rows are sorted upon their lexical ordering
- The `CIFLoops` are sorted alphabetically using the name of their first column.

- `void processSu (boolean b)`

Determines whether numeric variables with esds in brackets should be parsed.

To illustrate this further an example of the canonicalization algorithm for a small set of CIF data is given in Table 3.4.4.

### 3.4.5 Using CIFXML-J

CIFXML-J is a toolkit and can be used for many purposes. As all `CIFXML-J` elements are subclassed from the `XOM Element` class, `CIFXML-J` uses many XML functions from the `XOM` library.

#### Implementations

Each of the CIF classes has an API to facilitate the programmatic adding, removing, setting and getting of its particular data fields. For example, some of the methods of `CIFItem` are:

Before canonicalization	After canonicalization
data_xY  ITEM2 1.2  loop _col2 _col1 99 4 101 3  _item1 1.1  loop _zzz _aaa 1 z 99 q	data_xY  _item1 1.1 _item2 1.2  loop _aaa _zzz z 1 q 99  loop _col1 _col2 4 99 3 101

Table 3.1: Example of applying the canonicalization algorithm

```

/** set the name for a data item.
 * normally used when building a CIFXML.
 * data names should never be reset
 * implementers may check the value of a name or whether it
 * violates any CIF syntax or dictionary restrictions.
 * @param name (should be compliant with CIF syntax)
 * @throws CIFException syntax violation or ontology/dictionary
 * violation
 */
public void setItemName(String name) throws CIFException;

/** set the value for a data item.
 * normally used when building a CIFXML.
 * implementers may check the value to see whether it violates
 * any CIF syntax or dictionary restrictions.
 * @param value (should be compliant with CIF syntax). No
 * quotes are permitted unless part of the value
 * @throws CIFException syntax violation or ontology/dictionary
 * violation
 */
public void setItemValue(String value) throws CIFException;

/** get the name for a data item.
 * @return the name (should never be null)
 */
public String getItemName();

/** get the value for a data item.
 * @return the value
 */
public String getItemValue();

/** get the standard uncertainty for a data item.

```



```

* CIF parsers should ensure that if SU is non-blank then
* the data value should not contain a bracketed SU.
*@return the SU (null if not present).
*/
public Double getSU();

```

## Parsing, callbacks and DOM

CIFXML-J has a default parsing system which can be subclassed should a different parsing mechanism be needed. This allows the implementer or user to choose between parsers (including at runtime) perhaps on the basis of speed or conformance. In practice most programmers will use the default.

The SAX strategy is that a Parser provides callbacks when lexical/document events are fired. This means that the user delegates the parsing process to a Parser and only regains control after a complete parse (unless exceptions are thrown). The user provides callbacks to trap the events so that any that are not required can be ignored.

The following code is an excerpt from the `readToken` method of the `CIFParser` class, which shows a callback to the `CIFContentHandler` (`contentHandler`) to add a `CIFItem` (`item`) to the current instance of a `CIFDataBlock`. If there is an error during this method call, there is a callback to the `CIFErrorHandler` (`errorHandler`) to provide the error message.

```

ParserMessage m = contentHandler.addItem(item);
if (m != null) {
    errorHandler.error(m.getMessage(), this);
}

```

While CIFXML-J parsing is based on SAX, once the parsing is complete the in-memory representation of the CIFXML is a DOM (represented by an instance of the `CIF` class). It is then possible to manipulate this DOM using those CIFXML-J classes that represent the CIF elements (as described earlier).

## Example use of the CIFParser class

The following code will read a CIF into CIFXML, canonicalize it and then write out the CIFXML.

```
package sandbox;

import java.io.IOException;
import nu.xom.Document;
import uk.co.demon.ursus.cif.CIF;
import uk.co.demon.ursus.cif.CIFException;
import uk.co.demon.ursus.cif.CIFParser;

public class ParseIntoCifDom {
    public static void main(String[] args) {
        String filename = "C:/path/to/your/cif/file.cif";
        try {
            CIFParser parser = new CIFParser();
            // Set 'skip errors' option. Applies heuristics
            // telling the parser to fix or skip CIF
            // sections with recoverable errors.
            parser.setSkipErrors(true);
            // Set 'skip header' option. Tells the parser to
            // skip any comments that occur before the first
            // datablock.
            parser.setHeader(true);
            // Parse CIF into XML document using XOM
            Document doc = parser.parse(filename);
            // Root element is always a CIF object
            CIF cif = (CIF) doc.getRootElement();
            // Apply canonicalization algorithm to CIF object.
            cif.canonicalize();
            String outfile = "C:/path/to/write/to.xml";
            // Write CIFXML to path specified as 'outfile'
            FileWriter fw = new FileWriter(outfile);
            cif.writeXML(fw);
            fw.close();
        } catch (Exception e) {
            System.err.println("Error processing CIF file: '"+filename);
        }
    }
}
```

## A simple CIF editor

The following code shows how to read a CIF into CIFXML and to manipulate it through DOM-like calls, thus providing some of the features of a simple editing system. After creating the CIF, the process iterates over the datablocks and as an example manipulate the `_cell_measurement_temp` item. In this case it will either add a new item or change the value of the current one.

```
String file = "C:/path/to/your/cif/file.cif";
```

```

try {
    String tempItemName = "_cell_measurement_temperature";
    // values for the new temperature, standard uncertainty
    // and the number of decimal places to be used.
    double newTemp = 205.0;
    double newSu = 1.0;
    int dps = 1;
    // parse the CIF and get the root CIF element
    CIF cif = (CIF) new CIFParser().parse(file)
        .getRootElement();
    // for each datablock in the CIF
    for (CIFDataBlock block : cif.getDataBlockList()) {
        // try to find the cell measurement temperature item
        CIFItem temp = block.getChildItem(tempItemName);
        if (temp == null) {
            // if one doesn't exist then create a new one with
            // the values above, and add to the datablock
            temp = new CIFItem(tempItemName, newTemp,
                newSu, dps);
            block.appendChild(temp);
        } else {
            // if one does exist, then change its values to
            // those specified above
            temp.setValueAndSu(newTemp, newSu, dps);
        }
    }
    // write the CIF back out over the old file
    FileWriter fw = new FileWriter(file);
    cif.writeCIF(fw);
    fw.close();
} catch (Exception e) {
    System.err.println("Error processing CIF file: '"+filename)";
}

```

### 3.4.6 CIF to CIFXML conversion statistics

To investigate the conformance of published CIFs to the CIF standard, all CIF documents aggregated by CrystalEye (see section 4.3) between 2001-7 were parsed using CIFXML-J. The optional heuristics were used to fix minor deviations from the standard (see section 3.4.1), though more serious errors caused the parser to fail with controlled errors. Each CIF may contain more than one error, though here only the first unrecoverable error that the parser encountered is noted.

Figure 3.7 shows the percentage of parsing failures for CIFs from all publishers, highlighting the rapid decline in the publication of invalid CIFs. Figure 3.8 shows the percentage of failures per publisher (using only those publishers for which CIFs have been provided throughout the period). For CIFs from Acta Cryst. there were no errors throughout the entire period, while

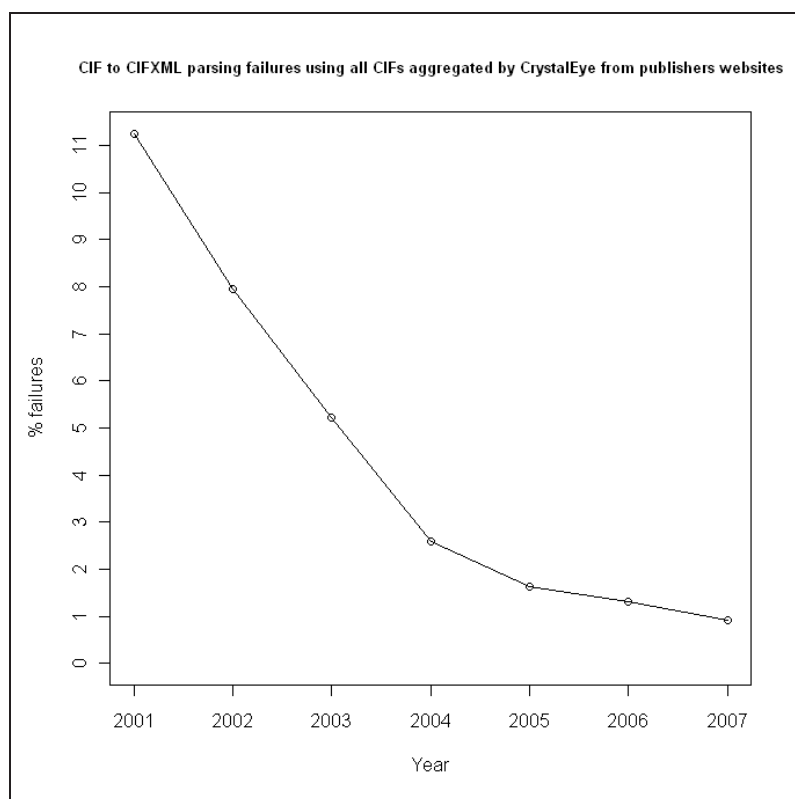


Figure 3.7: Plot showing the percentage of failures in parsing CIFs to CIFXML using all CIFs aggregated by CrystalEye from 2001-7.

RSC has had an almost 0% failure rate since 2004. There is still a higher proportion of failures with CIFs from ACS, though the rate is steadily decreasing. Presumably these effects are due to the greater use of checkCIF (which Acta Cryst. uses for every CIF it publishes), conformance in software and the greater familiarity with CIF in the editing processes.

By using the error messages produced by CIFXML-J and some manual inspection, the causes of errors were seen to fall into five categories (all of which could easily be avoided by submitting the CIF to checkCIF):

1. Incorrect use of delimiters
2. Absence of the first `data_` token

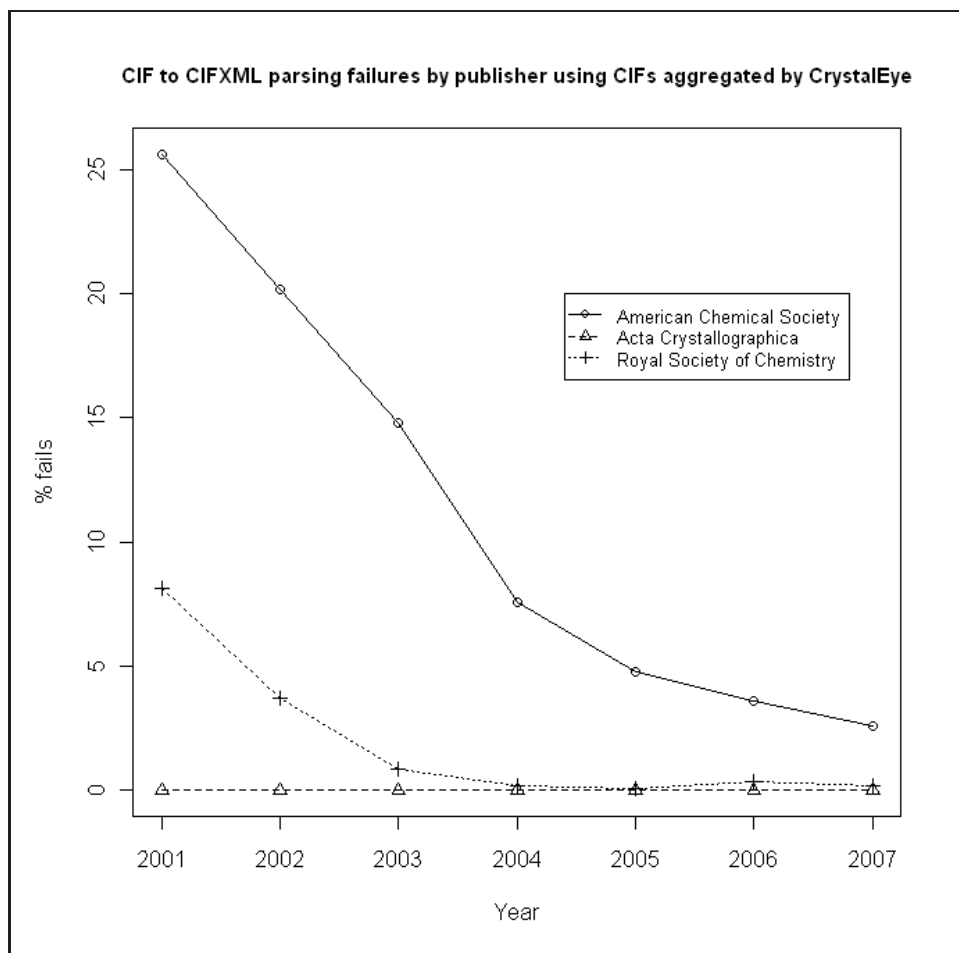


Figure 3.8: Plot showing the percentage of failures in parsing CIFs to CIFXML using CIFs aggregated by CrystalEye from ACS, Acta Cryst. and RSC from 2001-7.

3. Improper insertion of comments
4. Illegal characters
5. The number of items in a loop not being exactly divisible by the number of data names provided

Figures 3.9 and 3.10 show the number of each of these errors for each year from RSC and ACS journals respectively. The most common error from both sources is consistently the incorrect use of delimiters, though the proportion of invalid CIFs containing illegal characters has risen considerably throughout the period. It is noted that the majority of those loops with an invalid number of items were caused by the CIF being published in a truncated form.

### 3.5 CIFConverter: converting CIF to CML

The `legacy2cml` Java library is part of the CML project at Sourceforge[114] and provides functionality to convert several legacy chemical document formats into CML (*e.g.* MOL, SDF, ChemDraw CDX, MSDS). To allow the semantically lossless, dictionary validated conversion of CIFs into CML, I have added the `CIFConverter` class to the library. This section describes the process `CIFConverter` uses to convert CIF to CML before giving examples of its use in Java code.

The first step `CIFConverter` performs when invoked is use the `CIFXML-J` library to convert the supplied CIF into CIFXML. Thus, a CIF such as:

```
data_global
  _publ_contact_author_name      'A. Person'
  _publ_contact_author_address
;
University of Crystalville
Anycity
Anyland 1234
;

data_I
  _chemical_formula_sum          'As Cu S'
  _symmetry_space_group_name_H-M 'P n m a'

loop_
  _symmetry_equiv_pos_as_xyz
```

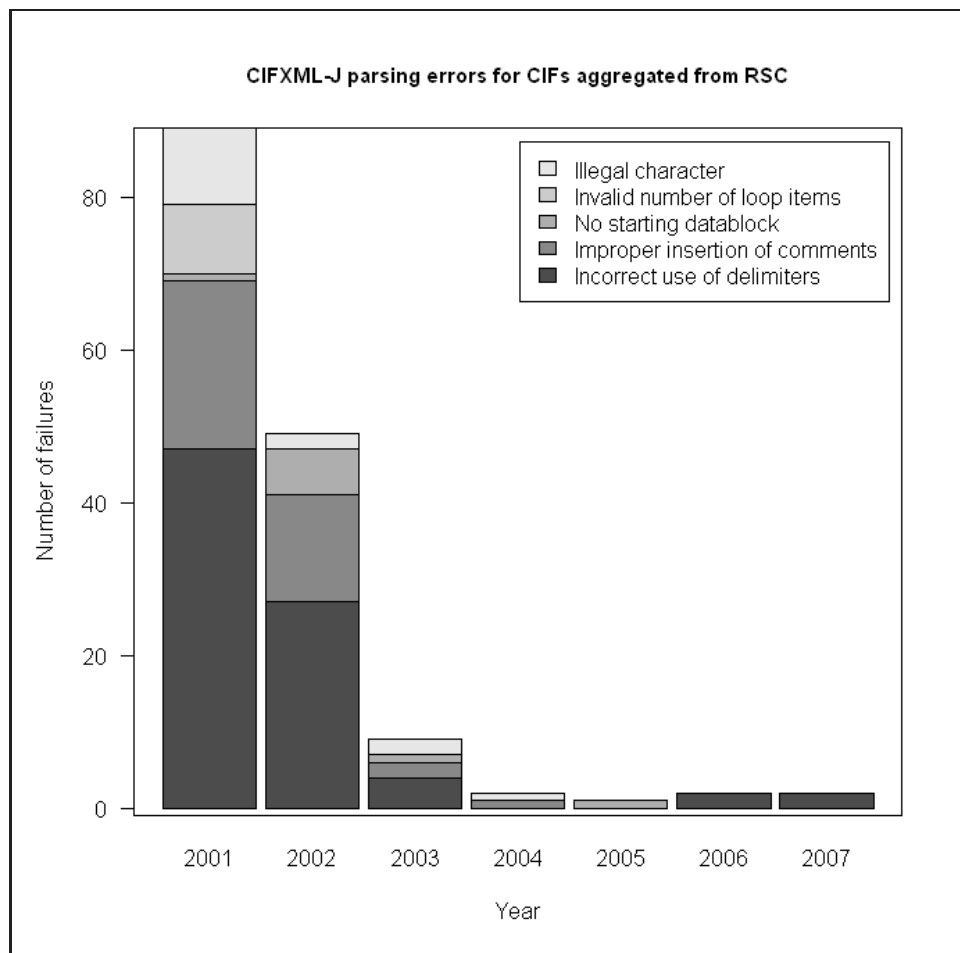


Figure 3.9: Number of CIFXML-J parsing errors per year by cause for RSC CIFs.

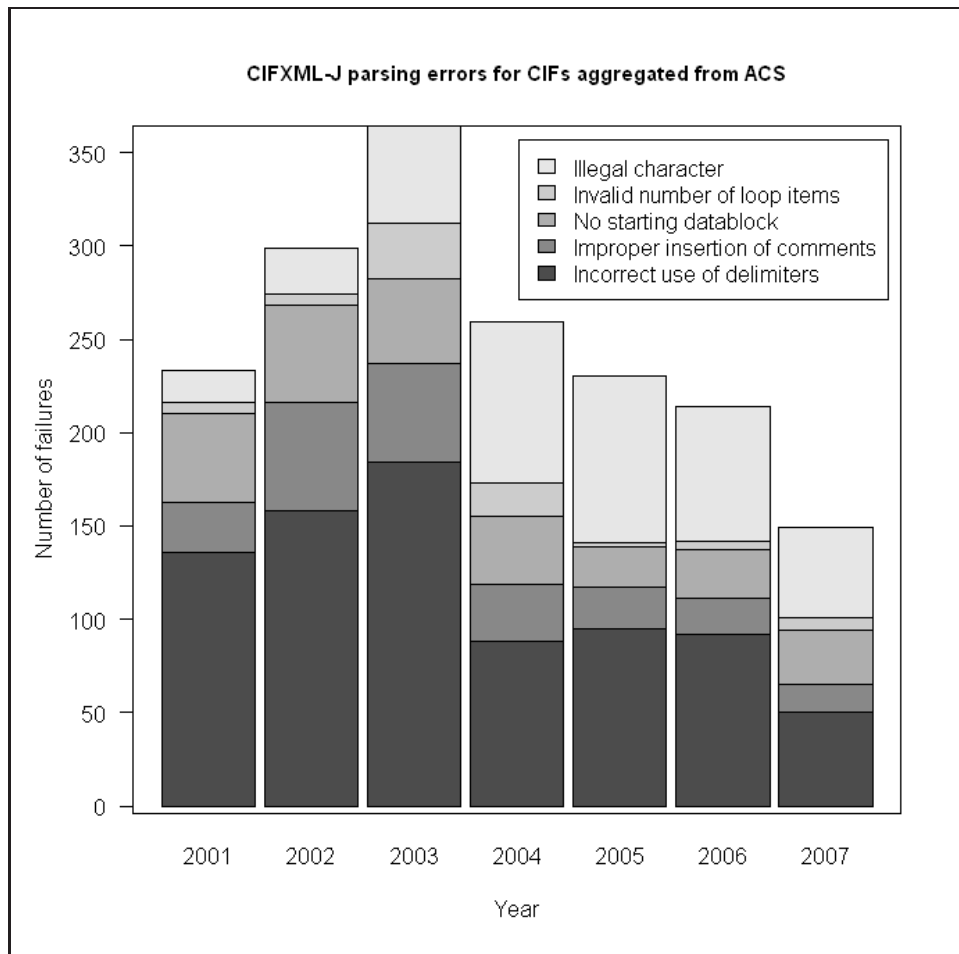


Figure 3.10: Number of CIFXML-J parsing errors per year by cause for ACS CIFs.



```

      'x, y, z'
      '-x+1/2, -y, z+1/2'
      'x+1/2, -y+1/2, -z+1/2'
      '-x, y+1/2, -z'
      '-x, -y, -z'
      'x-1/2, y, -z-1/2'
      '-x-1/2, y-1/2, z-1/2'
      'x, -y-1/2, z'
_cell_length_a      11.347(4)
_cell_length_b      3.7533(7)
_cell_length_c      5.4530(10)
_cell_angle_alpha    90.00
_cell_angle_beta     90.00
_cell_angle_gamma    90.00
_cell_formula_units_Z  4

loop_
  _atom_site_type_symbol
  _atom_site_label
  _atom_site_fract_x
  _atom_site_fract_y
  _atom_site_fract_z
  _atom_site_U_iso_or_equiv
  _atom_site_adp_type
  _atom_site_calc_flag
  _atom_site_refinement_flags
  _atom_site_occupancy
  _atom_site_disorder_assembly
  _atom_site_disorder_group
Cu Cu 0.17454(7) 0.2500 0.06264(18) 0.0165(2) Uani d S 1 . .
As As 0.01373(5) 0.2500 0.35177(11) 0.00894(18) Uani d S 1 . .
S S 0.16576(12) 0.7500 0.8196(3) 0.0100(3) Uani d S 1 . .
loop_
  _atom_site_aniso_label
  _atom_site_aniso_U_11
  _atom_site_aniso_U_22
  _atom_site_aniso_U_33
  _atom_site_aniso_U_12
  _atom_site_aniso_U_13
  _atom_site_aniso_U_23
Cu 0.0171(3) 0.0168(4) 0.0157(4) 0.000 -0.0015(3) 0.000
As 0.0105(3) 0.0084(3) 0.0080(3) 0.000 0.00083(17) 0.000
S 0.0108(5) 0.0114(5) 0.0079(5) 0.000 -0.0005(4) 0.000

```

is converted into an in-memory CIFXML DOM, which, when serialized would look as below (note that some sections have been truncated for brevity):

```

<cif>
  <datablock id="global">
    <item name="_publ_contact_author_name">A. Person</item>
    <item name="_publ_contact_author_address">
      University of Crystalville Anycity Anyland 1234
    </item>
  </datablock>
  <datablock id="I">
    <item name="_chemical_formula_sum">As Cu S</item>
    <item name="_symmetry_space_group_name_h-m">P n m a</item>
    <loop names="_symmetry_equiv_pos_as_xyz">
      <row>
        <cell>x, y, z</cell>

```

```

</row>
<row>
  <cell>-x+1/2, -y, z+1/2</cell>
</row>
...rest of rows...
</loop>
<item name="_cell_length_a">11.347(4)</item>
<item name="_cell_length_b">3.7533(7)</item>
<item name="_cell_length_c">5.4530(10)</item>
...rest of cell parameters...
<item name="_cell_formula_units_z">4</item>
<loop names="_atom_site_type_symbol
  _atom_site_label
  _atom_site_fract_x
  _atom_site_fract_y
  _atom_site_fract_z
  _atom_site_U_iso_or_equiv
  _atom_site_adp_type
  _atom_site_calc_flag
  _atom_site_refinement_flags
  _atom_site_occupancy
  _atom_site_disorder_assembly
  _atom_site_disorder_group">
  <row>
    <cell>Cu</cell>
    <cell>Cu</cell>
    <cell>0.17454(7)</cell>
    <cell>0.2500</cell>
    <cell>0.06264(18)</cell>
    <cell>0.0165(2)</cell>
    <cell>Uani</cell>
    <cell>d</cell>
    <cell>S</cell>
    <cell>1</cell>
    <cell>.</cell>
    <cell>.</cell>
  </row>
  ...rest of rows...
</loop>
<loop names="_atom_site_aniso_label
  _atom_site_aniso_U_11
  _atom_site_aniso_U_22
  _atom_site_aniso_U_33
  _atom_site_aniso_U_12
  _atom_site_aniso_U_13
  _atom_site_aniso_U_23">
  <row>
    <cell>Cu</cell>
    <cell>0.0171(3)</cell>
    <cell>0.0168(4)</cell>
    <cell>0.0157(4)</cell>
    <cell>0.000</cell>
    <cell>-0.0015(3)</cell>
    <cell>0.000</cell>
  </row>
  ...rest of rows...
</loop>
</datablock>
</cif>

```

The processing performed by `CIFConverter` is then done by acting on this DOM.

### 3.5.1 Handling multiple datablocks

The CIF example above contains two datablocks, one named `global` that contains researcher metadata and one named `I` which contains the crystal structure data and metadata. While not in the specification, this method of having the researcher metadata in one datablock (usually named `global`<sup>†</sup>) and then having one arbitrarily named datablock per crystal structure is used almost universally.

`CIFConverter`'s method of handling a CIF with more than one structure datablock is to produce one CML document for each. As there is just one `global` datablock, the data from this is copied to each of the created CML documents (as shown in figure 3.11). An example of the conversion of a CIF containing more than one structure datablock is contained within section 3.5.5.

### 3.5.2 Building the CML

After the CIFXML has been created, `CIFConverter` will then iterate through the child elements of each structure datablock and convert them into CML. This is subsequently merged with the CML created from the `global` datablock.

Where there is no direct semantic conversion from CIF to CML, references to the CIF dictionary are made through `dictRef` attributes. XML Schema data types are referenced through `dataType` attributes.

The CML created from the previous CIFXML document is (again, truncated for brevity):

---

<sup>†</sup>hence it will be referred to as the `global` datablock from now on.

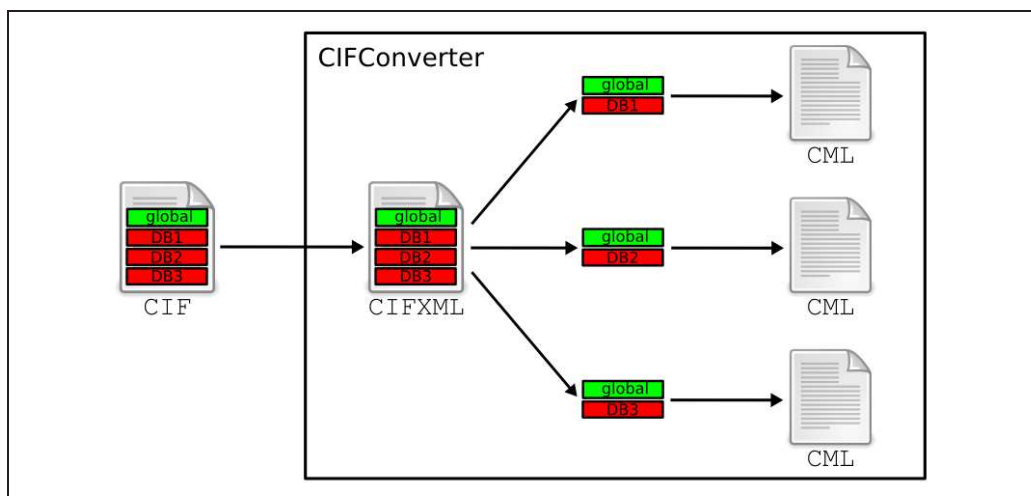


Figure 3.11: For each structure datablock (red) in the supplied CIF, CIFConverter will produce one CML document. Each of these will also contain the data from the CIF's global datablock (green).

```

<cml id="I" title="I" xmlns="http://www.xml-cml.org/schema">
  <scalar dictRef="iucr:_publ_contact_author_name" dataType="xsd:string">
    A. Person
  </scalar>
  <scalar dictRef="iucr:_publ_contact_author_address" dataType="xsd:string">
    University of Crystalville Anycity Anyland 1234
  </scalar>
  <formula concise="As 1 Cu 1 S 1"
    dictRef="iucr:_chemical_formula_sum" inline="As Cu S">
    <atomArray elementType="As Cu S" count="1.0 1.0 1.0"/>
  </formula>
  <molecule>
    <crystal z="4">
      <scalar dictRef="iucr:_cell_length_a"
        dataType="xsd:double" errorValue="0.0040">
        11.347
      </scalar>
      <scalar dictRef="iucr:_cell_length_b"
        dataType="xsd:double" errorValue="7.0E-4">
        3.7533
      </scalar>
      <scalar dictRef="iucr:_cell_length_c"
        dataType="xsd:double" errorValue="0.0010">
        5.453
      </scalar>
      ...rest of cell parameters...
    <symmetry spaceGroup="P n m a">
      <transform3>
        1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0
      </transform3>
      <transform3>
        -1.0 0.0 0.0 0.5 0.0 -1.0 0.0 0.0 0.0 0.0 1.0 0.5 0.0 0.0 0.0 1.0
      </transform3>
    </symmetry>
  </molecule>
</cml>

```

```

    ...rest of symmetry elements...
  </symmetry>
</crystal>
<atomArray>
  <atom id="a1" elementType="Cu" xFract="0.17454" yFract="0.25" zFract="0.06264"
    occupancy="1.0">
    <scalar dictRef="iucr:_atom_site_u_iso_or_equiv" dataType="xsd:double">
      0.0165
    </scalar>
    <scalar dictRef="iucr:_atom_site_adp_type" dataType="xsd:string">
      Uani
    </scalar>
    <scalar dictRef="iucr:_atom_site_label" dataType="xsd:string">
      Cu
    </scalar>
    <scalar dictRef="iucr:_atom_site_refinement_flags" dataType="xsd:string">
      S
    </scalar>
  </atom>
  ...rest of atoms...
</atomArray>
</molecule>
<table tableType="columnBased" dictRef="iucr:_atom_site_[]">
  <arrayList>
    <array dataType="xsd:string" dictRef="iucr:_atom_site_aniso_label" delimiter="|">
      |Cu|As|S|
    </array>
    <array dataType="xsd:double" dictRef="iucr:_atom_site_aniso_U_11" delimiter="|">
      |0.0171|0.0105|0.0108|
    </array>
    ...rest of arrays...
  </arrayList>
</table>
</cml>

```

## Creating the molecule

During the conversion, those CIFXML `items` and `loops` that describe the unit cell and the atom sites are brought together to create a CML `molecule`. By doing this, the distributed data structure inherited from the CIF is converted into a hierarchical structure, *i.e.* all data describing a particular atom site is contained as attributes or child elements of a CML `atom`. As shown, the `molecule` has two child elements:

- `crystal` — contains an attribute created from the `_cell_formula_units_z` item, plus the following child elements:
  - six `scalar` elements — representing the cell parameters, created from the `_cell_length_[]` and `_cell_angle_[]` items,

- a `symmetry` element — which has an attribute created from the `_symmetry_space_group_name_h-m` item and child `transform3` elements to represent the 4×4 matrix equivalents of the cell symmetry elements in the `_symmetry_equiv_pos_as_xyz` loop.
- `atomArray` — contains `atoms` created from the data in the `_atom_site_[]` loop.

Note that data with the *inapplicable* value (see point 23 in Common Semantic Features of the CIF specification[117]) not in a `table` will be omitted from the CML output (shown by the data for `_atom_site_disorder_assembly` in the above CIF).

The chemical validation steps performed during this conversion are:

1. checking element types are valid,
2. asserting that the six cell parameters have been provided,
3. checking that the symmetry elements and/or space group have been provided.

If any of these checks fail there is no way to confidently construct the crystal from the data provided, and an error will be thrown. If the unit cell symmetry elements are missing, then they can be deduced from the space group symbol, as described in the next subsection.

### Converting everything else

The other `items` and `loops` are processed in the following way:

- `_chemical_formula_[]` – converted into a CML `formula` element, *e.g.* the `_chemical_formula_sum` item in the above example,
- `items` – converted into `scalar` elements,
- `loops` – converted into `table` elements.

Note that all `formula`, `scalar` and `table` elements have the appropriate dictionary and data type reference attributes.

### 3.5.3 Adding symmetry elements

The symmetry elements (as provided by the `_symmetry_equiv_pos_as_xyz` data item) are essential for a complete description of a unit cell. They are necessary, along with the atom site descriptions and cell parameters to reconstruct the unit cell or create the unique moieties contained within the crystal (as discussed in section 3.6). Whilst it is unlikely that a CIF would be published with these items missing, this is the case with many unpublished or pre-published CIFs that have been deposited in crystallographic databases. For instance, many structures in the Crystallography Open Database do not have symmetry elements.

To allow the optional addition of missing symmetry elements into CML during the `CIFConverter` process, an index of Hermann-Mauguin (H-M) space group symbols against the corresponding symmetry elements is provided with `legacy2cml`. This index was created from all post-2002 CIFs (to ensure high quality of data) that were aggregated from IUCr, RSC and ACS journals in the CrystalEye system (see section 4.3). Thus, the index is not a complete representation of all possible H-M symbols, but covers the most common examples. When such a CIF is processed, the symmetry elements will be added by comparing the H-M symbol against the index (if the H-M symbol is missing, then nothing can be done and an error is thrown).

Since this index was created, the Blue Obelisk[118] Data Repository[119] has provided an XML file relating all possible H-M symbols to their symmetry elements. In the future this index will replace the one that is currently distributed with `legacy2cml`.

### 3.5.4 Dictionary validation

`CIFConverter` provides optional dictionary validation during processing. To implement this, a CML representation of the coreCIF dictionary has been created; this is distributed with the `legacy2cml` library. A section of coreCIF in both CIF and CML is shown below:

## coreCIF (in CIF)

```
...
data_atom_site_fract_
  loop_ _name          '_atom_site_fract_x'
                        '_atom_site_fract_y'
                        '_atom_site_fract_z'

  _category            atom_site
  _type                numb
  _type_conditions     esd
  _related_item        '_atom_site_Cartn_'
  _related_function     alternate
  _list                yes
  _list_reference      '_atom_site_label'
  _definition
;          Atom-site coordinates as fractions of the _cell_length_ values.
;
...
```

## coreCIF (in CML)

```
<dictionary namespace="http://www.iucr.org/cif" xmlns="http://www.xml-cml.org/schema">
...
  <entry dataType="xsd:double" id="_atom_site_fract_x">
    <scalar dictRef="iucr:list">yes</scalar>
    <definition>
      Atom-site coordinates as fractions of the _cell_length_ values.
    </definition>
    <scalar dictRef="iucr:category">atom_site</scalar>
  </entry>
  <entry dataType="xsd:double" id="_atom_site_fract_y">
    <scalar dictRef="iucr:list">yes</scalar>
    <definition>
      Atom-site coordinates as fractions of the _cell_length_ values.
    </definition>
    <scalar dictRef="iucr:category">atom_site</scalar>
  </entry>
  <entry dataType="xsd:double" id="_atom_site_fract_z">
    <scalar dictRef="iucr:list">yes</scalar>
    <definition>
      Atom-site coordinates as fractions of the _cell_length_ values.
    </definition>
    <scalar dictRef="iucr:category">atom_site</scalar>
  </entry>
...
</dictionary>
```

During dictionary validation, each time a data item is processed, the existence of the data name is checked, as well as the type of the corresponding data value. As the dictionary is in an XML format, existence of a data name can be checked by performing a simple XPath expression, such as:

```
./cml:dictionary/cml:entry[@id='_atom_site_fract_x']
```



which would check for an entry with the `_atom_site_fract_x` name. The data type can then be checked by inspecting the `dataType` attribute on the returned `entry`. If the `entry` is missing or the data is of the wrong type, then depending on the options set for `CIFConverter`, either an error will be thrown, or the data item will be discarded (with a warning message passed to `stderr`<sup>‡</sup>). Two warnings generated by `CIFConverter` are shown below:

```
Cannot parse _diffrn_standards_decay_% as double: 1.185 (random)
Cannot find dictionary item: _vrf_plat_213_i
```

The first indicates that the value should be a double, but cannot be parsed as such (because the text `(random)` has been included). The second indicates that the data name is not part of the dictionary provided for validation.

### 3.5.5 Using CIFConverter

The behaviour of `CIFConverter` may be controlled by command-line options (which may be passed to the class programmatically). The code below shows typical `CIFConverter` usage:

```
package ned24.sandbox;

import org.xmlcml.cml.legacy2cml.cif.CIFConverter;

public class CIFConverterTest {

    public static void main(String[] args) {
        String infile = "e:/folder/test.cif";
        String outfile = "e:/folder/test.cml";
        String cifDict = "e:/folder/dict/cifCoreDict.xml";
        String spaceGroupXml = "e:/folder/spaceGroup.xml";
        String[] args0 = {"-INFILE", infile,
            "-OUTFILE", outfile,
            "-SKIPERRORS",
            "-SKIPHEADER",
            "-SPACEGROUP", spaceGroupXml,
            "-DICT", cifDict
        };
        CIFConverter cifConverter = new CIFConverter();
        try {
            cifConverter.runCommands(args0);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

---

<sup>‡</sup>In Unix and Unix-like operating systems, as well as certain programming language interfaces, the standard streams are preconnected input and output channels between a computer program and its environment (typically a text terminal) when it begins execution. The three I/O connections are called standard input (*stdin*), standard output (*stdout*) and standard error (*stderr*). Standard error is an output stream typically used by programs to output error messages or diagnostics.

```

        } catch (Exception e) {
            System.err.println("Error processing CIF file: '"+infile);
        }
    }
}

```

The options that are set are:

- **INFILE** — the path of the CIF which is to be converted,
- **OUTFILE** — the path to which the output CML file will be written,
- **SKIPERRORS** and **SKIPHEADER** — options passed to the **CIFXML-J CIFParser** class so that errors during the CIFXML creation are fixed or skipped depending on their severity,
- **SPACEGROUP** — the path to the index containing the H-M symbols against the symmetry elements,
- **DICT** — the path to the dictionary file to be used.

For the above example, if the supplied CIF contains one structure datablock, then the resulting CML will be written to:

```
e:/folder/test.cml
```

If the CIF contains more than one structure datablock, then more than one CML file will be output, as described in section 3.5.1. These cannot all be written to the same location so an incremented integer is added to the supplied file name before the MIME type (in this case **.cml**). Thus, for a CIF with three structure datablocks, the following files will be written:

```

e:/folder/test_1.cml
e:/folder/test_2.cml
e:/folder/test_3.cml

```

### 3.5.6 CIFXML to CML conversion statistics

To investigate the completeness and chemical correctness of published CIFs, all those from ACS, Acta Cryst. and RSC that were correctly parsed using CIFXML-J in section 3.4.6 were processed and dictionary validated with CIFConverter. Figure 3.7 shows the percentage of parsing failures for CIFs from all publishers from 2001-7. The plot shows the decrease in failures during the period, and that since 2005, the failure rate has been below 0.2%, around an order of magnitude less than the corresponding figures for CIF to CIFXML conversion (see figure 3.7). It is worth noting that all of these errors were caused by data type errors or missing data from those sections of the CIF that are required to create a complete description of the unit cell, and that none were as a result of parsing or converting data from the CIFXML documents.

By using the error messages produced by CIFConverter and some manual inspection, the failure causing errors were seen to fall into six categories:

1. Non-IUPAC element symbol
2. Incomplete atom site data (generally the `_atom_site_type_symbol` is omitted, which is essential for obtaining the atom site element types)
3. Invalid atom site coordinates
4. No explicit atom data provided
5. Bad symmetry element description <sup>§</sup>
6. Incomplete cell parameters

The proportion of the failures occurring for ACS, Acta Cryst. and RSC were 59%, 13% and 28% respectively. The proportions of the occurrence of each error are shown in table 3.2.

---

<sup>§</sup>At present, if CIFConverter encounters a symmetry element in the `_symmetry_equiv_pos_as_xyz` loop that it is unable to parse, then an error is thrown. Though it has not yet been implemented, it would be possible for the parser to then find the correct set of symmetry elements using the provided space group symbol (as happens when no symmetry elements are provided).

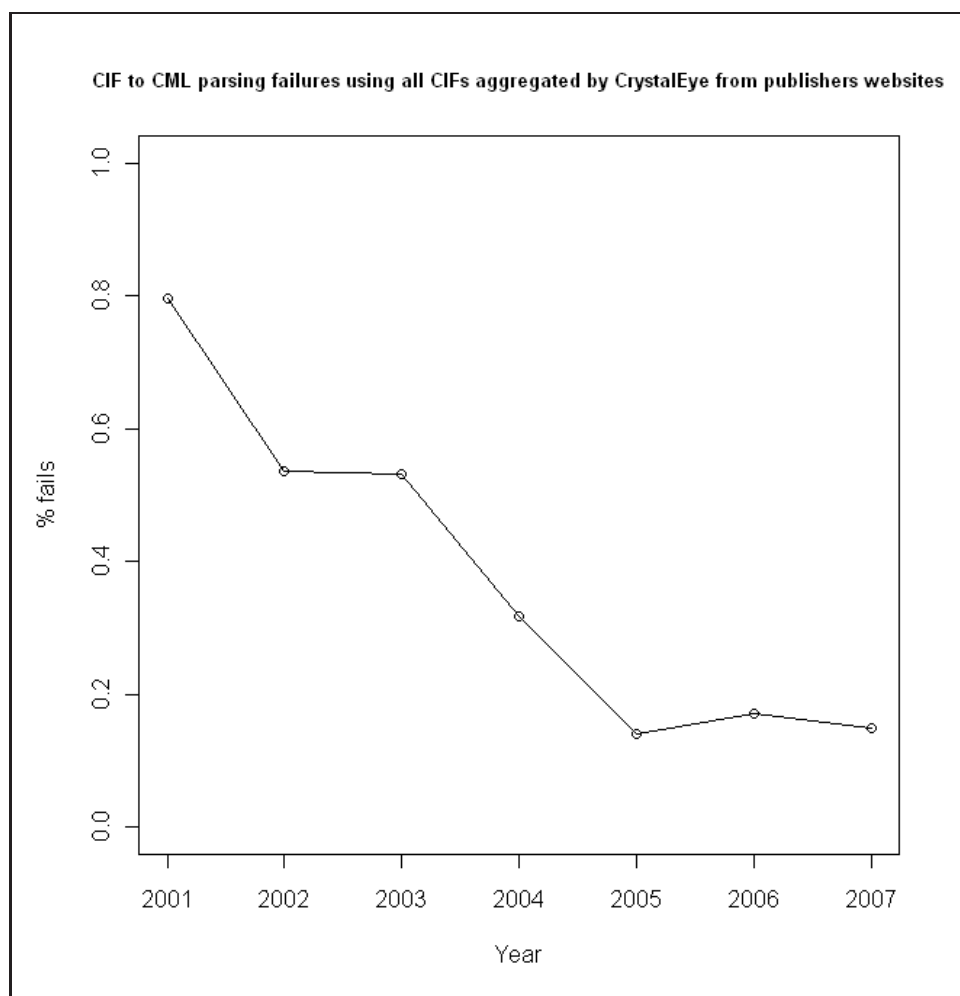


Figure 3.12: Plot showing the percentage of failures in converting CIFs to CML using all CIFs aggregated by CrystalEye from 2001-7.

Error	%
Non-IUPAC element symbol	62
Incomplete atom site data	4
Invalid atom site coordinates	9
No explicit atom data	8
Bad symmetry element	14
Incomplete cell parameters	3

Table 3.2: Proportion of error occurrences in CIF to CML conversion.

When using CIFConverter, if a CIF contains a non-IUPAC element symbol in an `_atom_site_type_symbol` item, then the parsing will always fail immediately. However, in the CIF standard this item is not limited only to valid element symbols, but may be:

...composed of any character except an underline... [122]

Common use shows that valid IUPAC symbols are almost universally applied to this item, though if a non-IUPAC symbol is entered (*e.g.* MN rather than Mn), checkCIF uses heuristics to discover the identity of the element, rather than suggest it be changed. This is presumably the cause of their relatively frequent occurrence. Owing to the above definition, if checkCIF’s heuristics cannot discover the identity of an element (*e.g.* J), then the error is only noted indirectly when there is a mismatch during cross-checking between the atom sites and any provided molecular mass or formula. Note that errors 2, 3, 4, 5 and 6 all give rise to A-level alerts when using checkCIF.

## 3.6 Enhancing the CML

To fully describe a crystal structure a CIF contains the unit cell parameters, symmetry elements and the general positions of the atom sites (see figure 3.13)<sup>¶</sup>. Connection tables for the moieties that compose the crystal structure are not given, though they are essential for associating identifiers to, and hence reusing the data. The `_chemical_conn_[]` data items have been defined in the CIF specification to allow a chemical connection table to be described.

<sup>¶</sup>In this section, all rendering of the CML data will be done using the Jmol application.

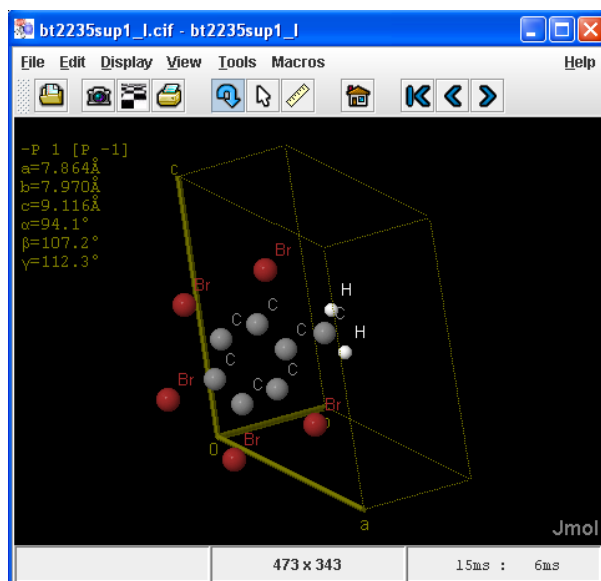


Figure 3.13: Rendering of the structural data in the CIF for 1,1'-ethane-1,2-diylbis(pentabromobenzene)[123]. The atom general positions, the cell parameters and symmetry elements (the P -1 space group contains the identity and a centre of inversion) are used to generate the complete unit cell.

However, these are not required for publication and as a result are not in common use. In all of the CIFs aggregated as part of the CrystalEye project (see section 4.3), none contained any of this category of data items.

This section describes software I created to enhance the data contained in CIFConverter output by:

- calculating the connection tables of the moieties in the crystal structure,
- converting these connection tables into unique chemical identifiers,
- merging this derived data with the original data.

The JUMBO library was an ideal starting point for this work as it provided the necessary base functionality for accessing and editing data in CML. All software described has now been added to the JUMBO library.

### 3.6.1 Creating the connection table

To find the connection table for each moiety, the following sequence is followed, starting with the CML output of `CIFConverter`:

1. discard minor disordered atom sites,
2. convert atom site fractional coordinates to cartesian,
3. add bonds between atoms using covalent radii,
4. apply symmetry operations to generate the molecular skeletons,
5. generate a set of bond orders and charges for the skeletons,
6. identify stereogenic atoms and bonds,
7. generate 2D coordinates for the molecules,
8. add 2D stereochemical details (*i.e.* wedge and hatch bonds).

This process is used for both discrete organic and organometallic moieties. For inorganic and polymeric organic or organometallic structures, as there is no obvious ‘simplest’ molecular unit, they are handled by adding all atoms to the unit cell. This is described further in the following subsections.

#### Processing disorder

In CIFs, disorder is defined through the `_atom_site_disorder_assembly` and `_atom_site_disorder_group` items, where the former is a

...code which identifies a cluster of atoms that show long-range positional disorder but are locally ordered. Within each such cluster of atoms, `_atom_site_disorder_group` is used to identify the sites that are simultaneously occupied.[120]

The `_atom_site_occupancy` item is used to report the fraction of the atom type present at a particular site. An example of a disordered methyl group in CIF is shown in figure 3.14.

loop_							
_atom_site_label				# *_assembly 'M' is a disordered methyl			
_atom_site_occupancy				# with configurations 'A' and 'B':			
_atom_site_disorder_assembly				#			
_atom_site_disorder_group				# H11B H11A H13B			
				# .   .			
C1	1	.	.	# .   .			
H11A	.5	M	A	# .   .			
H12A	.5	M	A	# C1 -----C2---			
H13A	.5	M	A	# / . \			
H11B	.5	M	B	# / . \			
H12B	.5	M	B	# / . \			
H13B	.5	M	B	# H12A H12B H13A			

Figure 3.14: A disordered methyl group represented in CIF (note that in CIF ‘#’ indicates a comment). There are two **groups** of three Hydrogen atoms (labelled A and B) belonging to the same **assembly** (labelled M). Each **group** is present 50% of the time.

If the disorder information is provided as described in the CIF specification, then discarding the minor disordered atom sites is simply a case of removing those that belong to **groups** with the lowest occupancies in each **assembly** (see figure 3.15). If there are two **groups** in an **assembly** with the same occupancy, then one will be chosen at random.

However, while occupancy data is always included in published CIFs, it is common for the **assembly** and **group** data to be either unspecified, incomplete or invalid with respect to the specification. If the software detects invalid disorder representation then it will try to deduce an unambiguous set of **assemblies** and **groups** from the occupancy data. In the majority of cases this is possible, though if there are atom sites with an occupancy of 0.5 present, it is not possible to infer with absolute certainty which are the correct **groups** (see figure 3.16).

## Creating the molecular skeleton

If the crystal structure is inorganic, no molecular skeleton is created, instead the atom sites are transformed using the cell symmetry elements to create a complete unit cell (see figure 3.17).



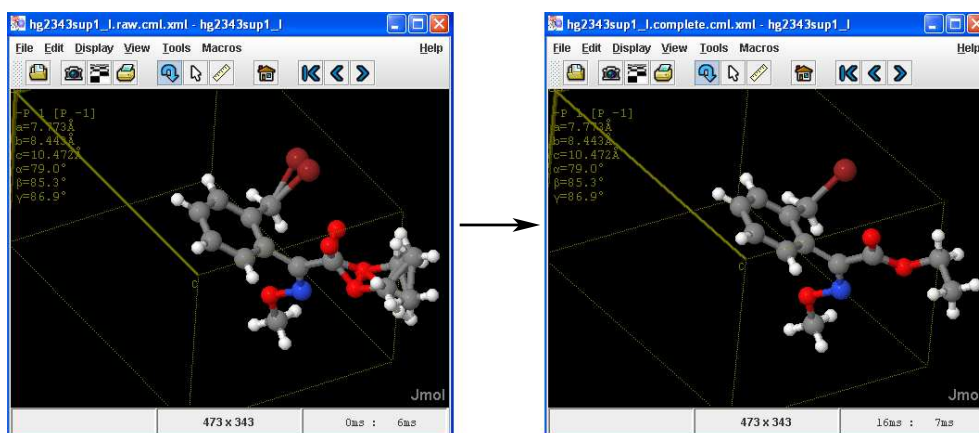


Figure 3.15: The removal of the minor disordered components from a crystal of (*E*)-Ethyl 2-[2-(bromomethyl)phenyl]-2-(methoxyimino)acetate[124], which includes disordered ester and bromine groups (the bonds have been included to aid clarity).

For organic and organometallic structures, the cell parameters and atom site fractional coordinates are used to calculate the cartesian coordinates for each atom[121]. From these, bonds between atoms are then added using ‘reasonable’ covalent radii, which results in one or more molecular fragments being generated. These fragments can then be transformed using the cell’s rotational symmetry elements, and bonds between the transformed and original fragments can be added to create the complete molecular skeletons (see figure 3.18).

From the molecular skeletons generated, it is then possible to detect if the structure is polymeric. If there are two or more symmetry related atoms in different environments, then the structure is taken as polymeric (as in figure 3.19).

For both inorganic and polymeric structures, creating the complete unit cell is the final stage in enhancing their CML. In addition, no further enhancement can be made to those moieties that have disorder that could not be resolved by the system. Note that those non-disordered moieties in an

```

loop_
  _atom_site_label
  _atom_site_fract_x
  _atom_site_fract_y
  _atom_site_fract_z
  _atom_site_U_iso_or_equiv
  _atom_site_adp_type
  _atom_site_calc_flag
  _atom_site_refinement_flags
  _atom_site_occupancy
  _atom_site_disorder_assembly
  _atom_site_disorder_group
  _atom_site_type_symbol
...
O2 0.5004(3) 0.6436(2) 0.7933(5) 0.0443(10) Uani d PD 0.55 Å -1 O
O3 0.5070(2) 0.5808(2) 1.0772(4) 0.0371(9) Uani d PD 0.55 Å -1 O
O4 0.4131(3) 0.5309(2) 0.8065(5) 0.0419(10) Uani d PD 0.55 Å -1 O
O5 0.6516(3) 0.5853(2) 1.3006(5) 0.0442(10) Uani d PD 0.55 Å -1 O
O6 0.6615(3) 0.5462(2) 0.9964(5) 0.0430(10) Uani d PD 0.55 Å -1 O
O2A 0.4256(3) 0.6081(3) 0.6883(5) 0.0418(12) Uani d PD 0.45 Å -1 O
O3A 0.5334(3) 0.5803(3) 0.9536(5) 0.0382(11) Uani d PD 0.45 Å -1 O
O4A 0.3763(3) 0.5199(2) 0.8929(6) 0.0355(10) Uani d PD 0.45 Å -1 O
O5A 0.6799(3) 0.6077(3) 1.1528(7) 0.0476(13) Uani d PD 0.45 Å -1 O
O6A 0.5367(3) 0.5670(3) 1.2664(6) 0.0467(13) Uani d PD 0.45 Å -1 O
...
...

loop_
  _atom_site_label
  _atom_site_fract_x
  _atom_site_fract_y
  _atom_site_fract_z
  _atom_site_U_iso_or_equiv
  _atom_site_adp_type
  _atom_site_calc_flag
  _atom_site_refinement_flags
  _atom_site_occupancy
  _atom_site_disorder_assembly
  _atom_site_disorder_group
  _atom_site_type_symbol
...
H23A -0.0344 1.0492 0.7977 0.119 Uiso calc PR 0.50 . . H
H23B 0.0543 0.9830 0.8921 0.119 Uiso calc PR 0.50 . . H
H23C -0.0673 0.8812 0.7592 0.119 Uiso calc PR 0.50 . . H
H23D 0.0028 0.8931 0.8349 0.119 Uiso calc PR 0.50 . . H
H23E -0.0859 0.9592 0.7405 0.119 Uiso calc PR 0.50 . . H
H23F 0.0357 1.0611 0.8735 0.119 Uiso calc PR 0.50 . . H
...

```

Figure 3.16: Two examples from published CIFs of `_atom_site_[]` loops which contain invalid disorder representation. The top example is invalid as it contains atom sites with occupancy < 1 that have a `group`, but not an `assembly`, as well as containing a `group` which is composed of atom sites with differing occupancies. This is resolvable, as the sites with the same occupancy can be assigned the same `group`, and the `groups` which total an occupancy of 1 can be assigned the same `assembly`. The bottom example has disordered atom sites with no assigned `group` or `assembly`. The sites all have the same occupancy, so this is unresolvable without inspecting the atomic geometry.

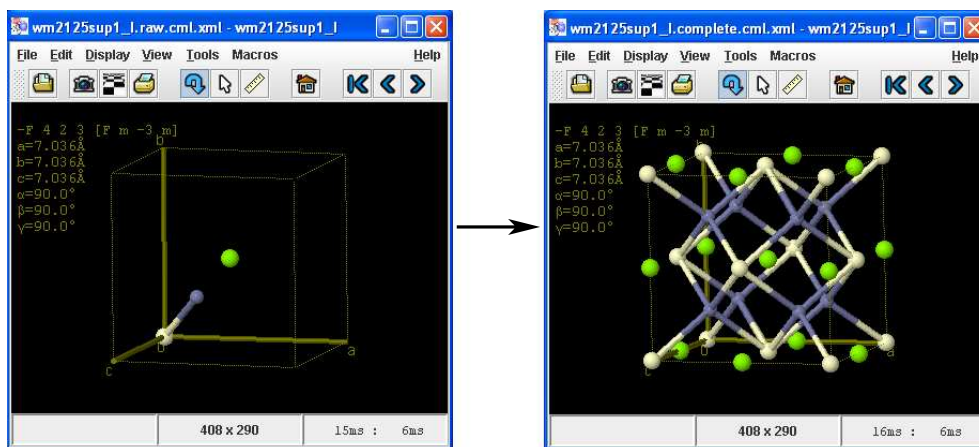


Figure 3.17: The creation of the complete unit cell for a crystal structure of  $\text{MgZn}_2\text{Ce}[125]$  (where Mg = green, Zn = white, Ce = grey).

unresolved disordered crystal are still processed further, as described in the following subsections.

### Adding bond orders and charges

As there are no concepts of bond orders or atomic charges in CIF, to create the complete connection table, the software must use a heuristic-based approach to deduce them from the molecular skeleton alone.

For commonly occurring molecules (*e.g.* solvents and some counter-ions), the system has a set of pre-defined templates which it will use to match their skeletons and then to add a set of bond orders and charges (BOACs). For all other organic moieties, the steps taken to find the correct set of BOACs are:

1. If available, obtain the moiety charge from the `_chemical_formula_moiety` item (*e.g.* `2(C17 H13 F3 N3 O +)`, `04 S 2-`, `2(H2 O)`). This is often not provided.
2. Add BOACs to common functional groups. Some groups, *e.g.* azides, quaternary ammonium cations and carboxylates can be unambiguously

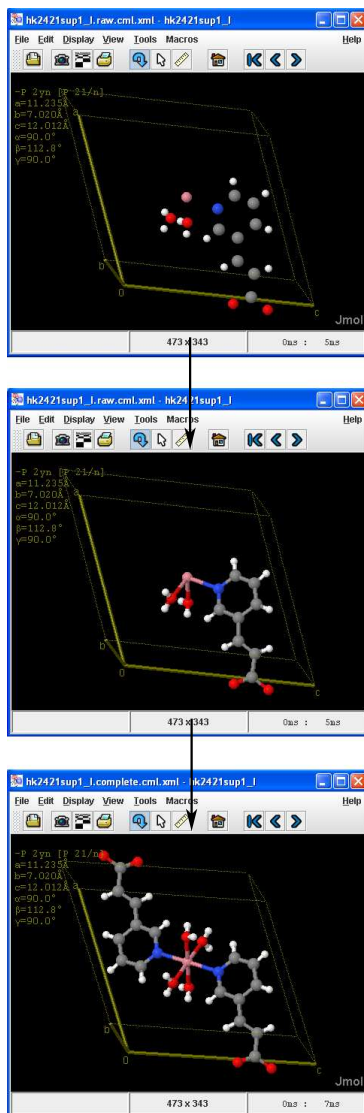


Figure 3.18: Images showing the steps taken to find the molecular skeleton for a *trans*-Tetraaquabis[3-(3-pyridyl)acrylato- $\kappa N$ ]cobalt(II) crystal[126]. The images show: (top) the general atom positions, (middle) the addition of bonds between atom sites to create a molecular fragment and (bottom) the generation of the complete molecular skeleton by transformation of that fragment using the cell symmetry and subsequent addition of bonds between the original and transformed fragments.

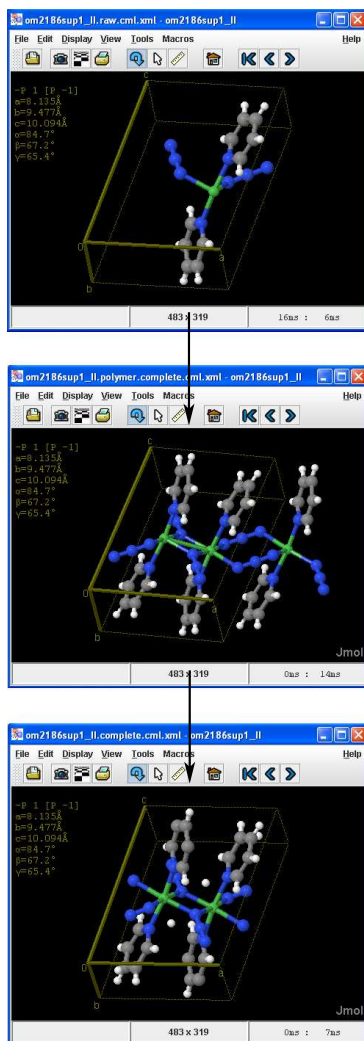


Figure 3.19: Images showing how the software detects and handles polymeric structures using a crystal structure of *catena*-Poly[[bis(pyridine- $\kappa N$ )nickel(II)]-di- $\mu$ -azido- $\kappa^4 N^1:N^3$ -[bis(pyridine- $\kappa N$ )nickel(II)]-di- $\mu$ -azido- $\kappa^4 N^1:N^1$ ][127]. The top image shows the fragment created after bonds have been added to the general atom positions, with the middle image showing the skeleton created after transforming and joining that fragment using the cell symmetry. In the middle image, the Nickel centre is shown to have three different environments, which indicates that the structure is incomplete (symmetry related centres in a crystal should have the same environment) and is polymeric. Upon detecting this, the software will represent the structure by adding all atoms to the unit cell (bottom).

identified by their skeleton, and so can be dealt with before the heuristic process begins.

3. Get the list of possibly charged heteroatoms which were not marked up in step 2, *e.g.* 2-coordinate Nitrogen may be negatively charged and 3-coordinate Nitrogen may be positively charged.
4. If the moiety charge was found in the first step, get all combinations of charged heteroatoms that give a total charge equal to the moiety charge. If the moiety charge was not found, then get all possible combinations of charged heteroatoms.
5. For each combination found in step 4, apply them to the molecular skeleton, and:
  - calculate the  $\pi$ -electrons on each atom (valence electrons minus ligands, *e.g.* a neutral carbon atom with two ligands has two (4-2)  $\pi$ -electrons),
  - add bonds between atoms containing  $\pi$ -electrons until no  $\pi$ -electrons remain, thus giving a complete set of BOACs for the skeleton.

If the moiety charge was provided in the first step, then any matching complete set of BOACs found can be confidently assigned to the molecular skeleton. If more than one complete set of BOACs is found, then the one containing the fewest charged atoms is selected.

If the moiety charge is not provided in the first step, then it must be deduced by the system before a set of BOACs can be added to the moiety.

- If the unit cell contains one moiety, then it can be assumed to have a charge of 0 (to give a neutral unit cell),
- If the unit cell contains more than one moiety, then for each, all steps up to this point must be run to find the complete sets of

(a)	Moiety	A	B	C
	Possible	0	0	+1
	Charges	-2	+1	

(b)	Moiety	A	B	C
	Possible	0	0	0
	Charges	-2	+1	+1

Figure 3.20: If a unit cell contains multiple moieties (in this case three, A, B and C), and no moiety charges have been provided, then all charges that give a complete set of BOACs for each moiety must be calculated. They are then compared to find a combination that gives a total charge of 0. In (a), there is one possible combination of moiety charges where  $[A=-2, B=+1, C=+1]$ . These can therefore be assigned to the moieties with confidence. In (b) there are two possible combinations  $[A=B=C=0]$  and  $[A=-2, B=+1, C=+1]$ , and the software is unable to deduce which is correct. In this situation, it is better to do nothing than risk introducing errors into the system, and so no BOACs will be added to the moiety skeletons.

BOACs that are feasible. The possible charges for each moiety are then compared to find a combination where the total charge for all moieties equals 0 (as described in figure 3.20).

6. If no complete sets of BOACS have been found, then steps 3 and 4 will be retried with anionic and cationic carbon also considered. This step is only used if the charge on the moiety is known. Otherwise, the number of possible combinations of charged atoms to inspect increases by orders of magnitude, making the process very slow.

An example of this process is shown in figures 3.21.

The process of adding BOACs to organometallic moieties is more difficult. Even if the `_chemical_formula_moiety` item is provided, it only gives the overall moiety charge for an organometallic species, and not the charges on any metal centres or organic ligands within. The metal charges can be found by parsing the chemical name provided in the `_chemical_name_systematic` item, *e.g.*

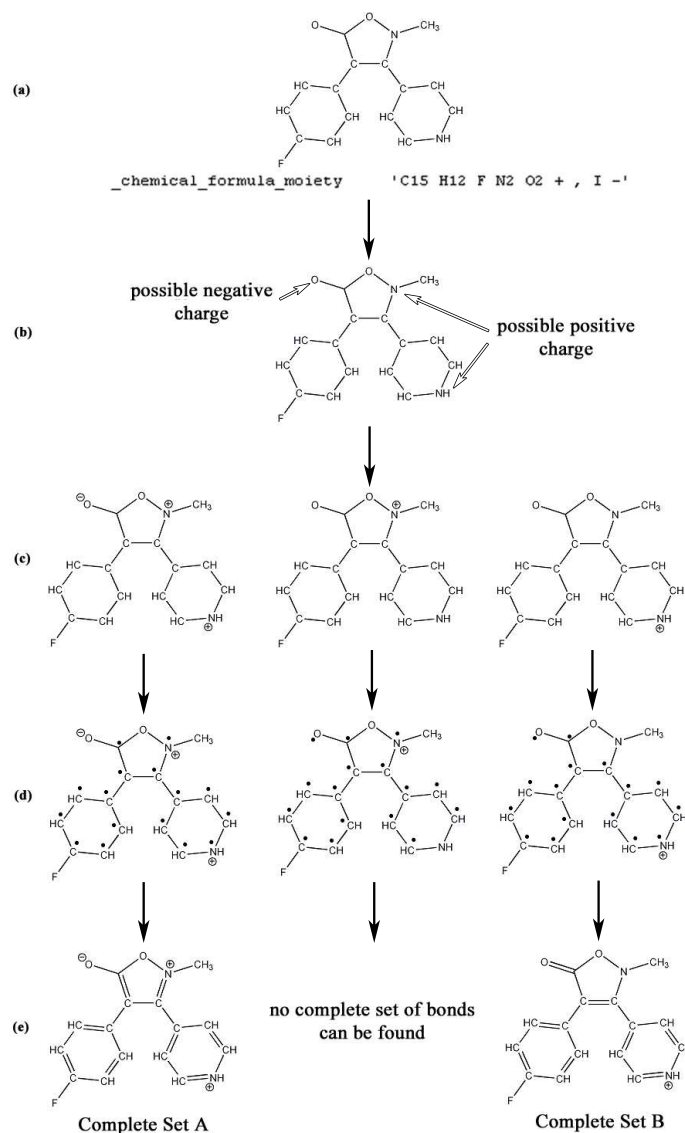


Figure 3.21: The process of adding bond orders and charges to a molecular skeleton, for which the moiety charge has been provided (a). There are no common functional groups recognised by the system, so the next step is to list the possibly charged heteroatoms (b). This is followed by creating molecular skeletons containing the charged heteroatom combinations that give a total moiety charge equal to that provided (c). For each, the  $\pi$ -electrons on each atom are calculated (d) before finding a set of bond orders which use all those  $\pi$ -electrons (e). This final stage gives the complete sets of BOACs for the moiety, Set A and Set B, which represent different resonance forms of the same structure. The set of BOACs with fewest charged atoms is selected (Set B) and applied to the moiety.



```

_chemical_name_systematic
;
Bis[2-(2-aminoethyl)-1H-benzimidazole-\k^2N^2^,N^3^]zinc(II) bis(perchlorate)
;

```

though this is not commonly available. When trying to find a set of BOACs for an organometallic moiety, there are three common scenarios:

1. **the moiety charge and metal charge(s) are known** – Using the same trial-and-error method previously described for organic moieties with no charge provided, all possible complete sets of BOACs are calculated for each ligand. The charges for those sets can then be compared to find an unambiguous combination that is equal to the difference between the overall moiety and metal charges provided. An example is shown in figure 3.22.
2. **the moiety charge is known** – Here the only way to be confident of adding BOACs for the whole moiety is if each ligand has only one possible set of BOACs, and there is only one metal centre (an example of adding BOACs to a multi-centre moiety is shown in figure 3.23). If that is the case, then the metal charge can be deduced as the difference between the provided moiety charge and the combined charges for all the ligands.
3. **no charges are known** – In this case, if there are multiple organometallic moieties, the metal charges can never be confidently deduced. As with multi-centre moieties in the previous point, it is possible to deduce the total charge on all metal centres in the cell if all ligands have only one possible set of BOACs, but it is not possible to then attribute a charge to a particular metal centre.

It is clear that there are going to be many organometallic structures for which the system cannot find an unambiguous set of BOACs. There are areas, such as this, in which computer programs may never be able to do the job of a trained chemist, and so other methods for completing the task must be considered. This is discussed further in section 4.3.8.



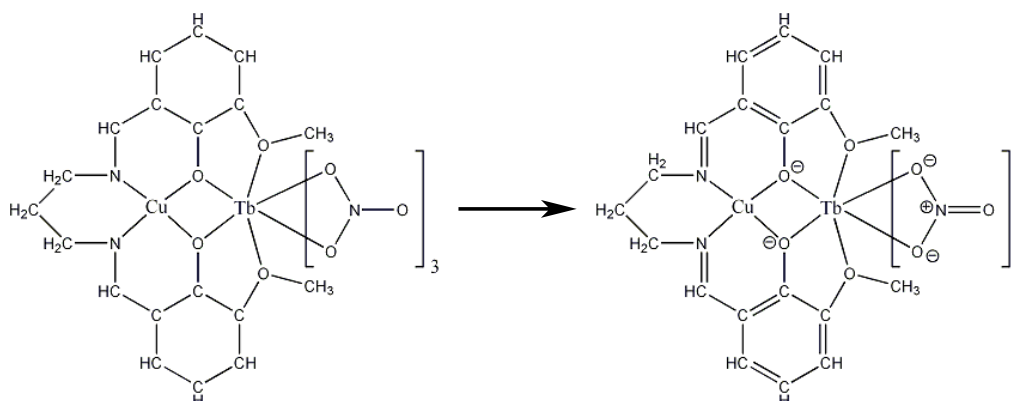


Figure 3.23: For the Copper-Terbium complex shown, the BOACs for the ligands can be unambiguously deduced from the skeleton. If the moiety charge is known to be 0, then the metal centres have a combined charge of +5. There is no way of knowing how to partition this charge correctly between the two centres, so no charges will be assigned.

## Adding stereochemistry

In those moieties that contain a complete set of BOACs, both 2D and 3D stereochemistry is then added. For 3D stereochemistry the following is performed:

- Stereogenic atoms – Each atom in the moiety is inspected for chirality, and for each chiral atom found, an atom parity is calculated and appended as a child element, *e.g.*

```
<atom id="a7" ... >
...
  <atomParity atomRefs4="a2 a16 a17 a8">10.524352510236906</atomParity>
</atom>
```

A positive parity is equivalent to the Cahn-Ingold-Prelog *R*, and a negative parity is equivalent to *S*.

- Stereogenic bonds – Each acyclic double bond is inspected for geometrical stereoisomerism. For each found, its configuration is calculated (*cis* or *trans*), and then added to the **bond** as a **bondStereo** child element. For instance:

```
<bond id="a54_a55" atomRefs2="a54 a55" order="2">
...
</bond>
```

may be described as a trans bond

```
<bond id="a54_a55" atomRefs2="a54 a55" order="2">
...
<bondStereo atomRefs4="a53 a54 a55 a57">T</bondStereo>
</bond>
```

To add 2D stereochemical descriptors, the 2D coordinates for each atom are first generated from the 3D coordinates using the CDK[39]. These are then added as attributes to the `atoms`, for instance:

```
<atom id="a47" z3="2.979995156779009" y3="1.7497709400000003" x3="2.789870115152765" ...>
...
</atom>
```

becomes

```
<atom id="a47" z3="2.979995156779009" y3="1.7497709400000003" x3="2.789870115152765"
x2="3.4837837797857762" y2="5.357026407595942" ...>
...
</atom>
```

Using the 2D coordinates, bonds can then be described as either *wedge* or *hatch*. These are used for representing the stereochemistry in 2D diagrams of the structures (such as those described in section 4.3.3). For instance:

```
<bond id="a7_a8" atomRefs2="a7 a8" order="1">
...
</bond>
```

may be described as a wedge bond (H is used for hatch bonds).

```
<bond id="a7_a8" atomRefs2="a7 a8" userCyclic="CYCLIC" order="1">
...
<bondStereo>W</bondStereo>
</bond>
```

The addition of stereochemistry to a moiety completes the creation of the connection table.

### 3.6.2 Adding canonical identifiers

For those moieties with a complete connection table, two canonical identifiers are added to their CML molecules as child `identifier` elements:

1. InChI – to allow exact matching of the moiety. This is created from the CML molecules using the `jni-InChI` library[128], a Java Native Interface to the InChI C++ library, which allows it be called directly from Java code.
2. SMILES – to allow substructure searches of the moiety. This is created from the CML molecules using the CDK implementation of the SMILES algorithm.

Thus, the CML for a crystal structure containing one moiety will be:

```
<cml xmlns="http://www.xml-cml.org/schema">
...
<molecule id='I'>
...
  <identifier convention="cdk:smiles">
    moiety smiles string
  </identifier>
  <identifier convention="iupac:inchi">
    moiety inchi string
  </inchi>
</molecule>
</cml>
```

If a crystal structure contains more than one moiety, and each has a complete connection table, then identifiers representing the whole crystal will also be added as child elements to the parent `molecule`. The CML for a crystal structure containing two moieties will be:

```
<cml xmlns="http://www.xml-cml.org/schema">
...
<molecule id='I'>
...
  <molecule id='I_sub1'>
    ...
    <identifier convention="cdk:smiles">
      moiety 1 smiles string
    </identifier>
    <identifier convention="iupac:inchi">
      moiety 1 inchi string
    </inchi>
  </molecule>
  <molecule id='I_sub2'>
```

```

...
<identifier convention="cdk:smiles">
  moiety 2 smiles string
</identifier>
<identifier convention="iupac:inchi">
  moiety 2 inchi string
</inchi>
</molecule>
<identifier convention="cdk:smiles">
  overall crystal smiles string
</identifier>
<identifier convention="iupac:inchi">
  overall crystal inchi string
</inchi>
</molecule>
</cml>

```

### 3.7 Current usage

The process of converting CIF to ‘enhanced’ CML has already been implemented in the following works:

- The CrystalEye[129] website (see chapter 4). This has exposed the process to CIFs from a wide range of laboratories with varying degrees of conformance to the exact standard.
- The SPECTRa project[130], in which the process was contained in repository software implemented at Cambridge University, Imperial College London and Southampton University.

### 3.8 Conclusions

It is possible for a machine to read and extract data from published CIFs with high degree of accuracy. The ability to create software to perform such processing is aided greatly by the provision of Open specifications and dictionaries, as is the case with CIF. The high rate of validity of published CIFs can be attributed to improved CIF editors and the increasing number of journals which require the use of checkCIF before publishing.

The generation of chemical connection tables from CIF data relies on a number of heuristics, though in many cases the correct representation can

be deduced. As shown, the calculation of the CT takes up a large part of the CIF-to-CML conversion, though it is important to note that all of this processing would be redundant if the `_chemical_conn` data items were required when publishing CIFs.

## Chapter 4

# Automating the aggregation, creation and dissemination of semantic crystallography

### 4.1 Introduction

As described in the previous chapter, CIFs have been adopted by many publishers as a standard method of publishing crystallographic data. When an article describing a crystal structure is submitted for publication, it is expected that the CIF will be provided as supplemental data. For some publishers, these CIFs are made available through Open Access (*i.e.* not behind a toll-access barrier, see figure 4.1), such as

- International Union of Crystallography (IUCr)
- American Chemical Society (ACS)
- Chemical Society of Japan (CSJ)
- Royal Society of Chemistry (RSC)

As a CIF consists of a set of facts about the crystal structure it describes, and as facts are not copyrightable *per se*, these CIFs can be viewed as Open Data and available for download and reuse. There is an abundance of crystallographic data available in this way, but it is disperse and in a format which



inorganic compounds

html

pdf

abstract

cif

3d view

structure factors

supplementary materials

checkCIF

open access

Acta Cryst. (2008). E64, i16 [ doi:10.1107/S1600536808003267 ]

$\alpha$ -Lead tellurite from single-crystal data

V. E. Zavodnik, S. A. Ivanov and A. I. Stash

Online 6 February 2008

html

pdf

abstract

cif

3d view

structure factors

supplementary materials

checkCIF

open access

Acta Cryst. (2008). E64, i17 [ doi:10.1107/S1600536808003231 ]

A quaternary germanium(II) phosphate, Na[Ge<sub>4</sub>(PO<sub>4</sub>)<sub>3</sub>]

C.-S. Lee and S.-F. Weng

Online 6 February 2008

Figure 4.1: Two article summaries in the table of contents from Acta Crystallographica Section E. Note the link to the CIF in the line of links above the article titles.

few tools understand. The Cambridge Crystallographic Data Centre[131] (CCDC) maintains a large collection of data from CIF files (the Cambridge Structural Database[132][133] (CSD)), predominantly from published articles, though to gain access to all of the data a substantial fee is required. There are no Open collections of this data available, and thus no easy way to use it for automated data-driven science. This chapter discusses work done to provide a system which can *automatically*

- aggregate the latest published CIFs from the Web,
- convert the data into CML,
- provide services for dissemination of all aggregated and derived data.

The initial effort focussed around implementing the system using workflow software. After several months of work, it was decided that this was not the best model for the system, and the software created up to that point

was separated from the workflow tool and developed into a standalone Java application, CrystalEye. Both of these methodologies are discussed in the following sections.

## 4.2 Implementing a workflow system

As this method was eventually abandoned, discussion of this work will be brief. The workflow software will be described to highlight the original perceived benefits it afforded, as well as those flaws that led to the discontinuation of its use. Discussion of the workflows created will be minimal as the tools and flow of data are described in detail in section 4.3.

### 4.2.1 The Taverna Workbench

The Taverna Workbench[134]<sup>\*†</sup> is a Java-based, Open Source software tool for designing and executing workflows. Taverna has been developed under the *my*Grid project, part of the UK eScience program. It allows users to construct complex analysis workflows from components on both remote and local machines, run these workflows on their own data and visualise the results. Taverna is aimed primarily at the bioinformatics community, though the software is generic and can be used for any domain.

#### Workflows and components

A *workflow* is a set of components and relations between them used to define a complex process from simple building blocks. Relations may be in the form of data links, transferring information from the output of one component to the input of another, or in the form of control links (*e.g.* a temporal ordering) which state some conditions on the execution of a component.

---

<sup>\*</sup>Which from now on, I will refer to as Taverna.

<sup>†</sup>Note that at the time this work was undertaken, the latest version of Taverna was v1.3.2. Problems discussed concerning this version of the software may have been resolved in subsequent versions.

A *component* is a reusable building block which performs some well defined function within a process. Components may consume information (via a set of input ports) and may emit information (via output ports) and may be located on any computational resource accessible via the Internet or on the user's local workstation. The different types of component available are

*Web Service* : Taverna allows the creation of components from SOAP<sup>‡</sup>-based WSs. If a WS provides a WSDL (Web Services Description Language) file, Taverna can parse it and retrieve the required information for communicating with that service. The bioinformatics community provides thousands of WSs to allow programmatic access to information repositories and analysis tools (*e.g.* XEMBL[135], openBQS[136] and SoapLab analysis services[137]). Many of these are automatically found by Taverna and made available as components in its *available services* panel (see figure 4.2). At the time of this development, there were no known WSs provided by the cheminformatics community.

*LocalWorker* : components made from Java classes that implement the LocalWorker interface provided in the Taverna library. A number are provided with the distribution, though users can create their own collections as JAR files and import them into Taverna.

*Beanshell* : a Java scripting language[138] that allows you to include Java code directly into a workflow. The Beanshell code is interpreted when the workflow is run.

*String constant* : components that supply a constant string value as input to another component.

*Nested workflow* : components of this type represent and invoke another workflow. They can be useful to create black-box processors so users can share and implement complex workflows as single components without needing to know the inner workings.

---

<sup>‡</sup>SOAP once stood for 'Simple Object Access Protocol' but this acronym was dropped with Version 1.2 of the standard, as it was considered to be misleading.

A workflow can also possess input and output data entities. A workflow input can be considered to be a source processor which executes instantaneously and makes the input value available on its virtual output port. A workflow output can be considered as a sink processor which receives a value from its input port but never actually executes. By assigning a MIME type to output ports, the method used to render output data can be signalled to the Taverna GUI.

## The GUI

The Taverna GUI (see figure 4.2) is broken down into three sub-windows

*Advanced model explorer* : used to link components of the workflow, as well as providing execution constraints (*e.g.* indicate that a process should be retried a number of times before failing).

*Available services* : a simple browser of those services known to Taverna. This includes all components contained within the toolkits incorporated into Taverna, as well as those WSs whose addresses have been hardcoded into the system. It allows services to be easily imported into workflows via the advanced model explorer.

*Workflow diagram* : renders the workflow model in SVG. Figure 4.3 shows the key for the workflow rendering in the GUI.

There is also an *enactor invocation* window which pops up when a workflow is run. While the workflow is running the progress is shown as a diagram, and upon completion the contents of the workflow outputs will be rendered. It is also possible to view the inputs and outputs of each individual component of the workflow (see figure 4.4).

Once all desired processes have been described and made available as Taverna components, creating and running a workflow using the GUI is simple.

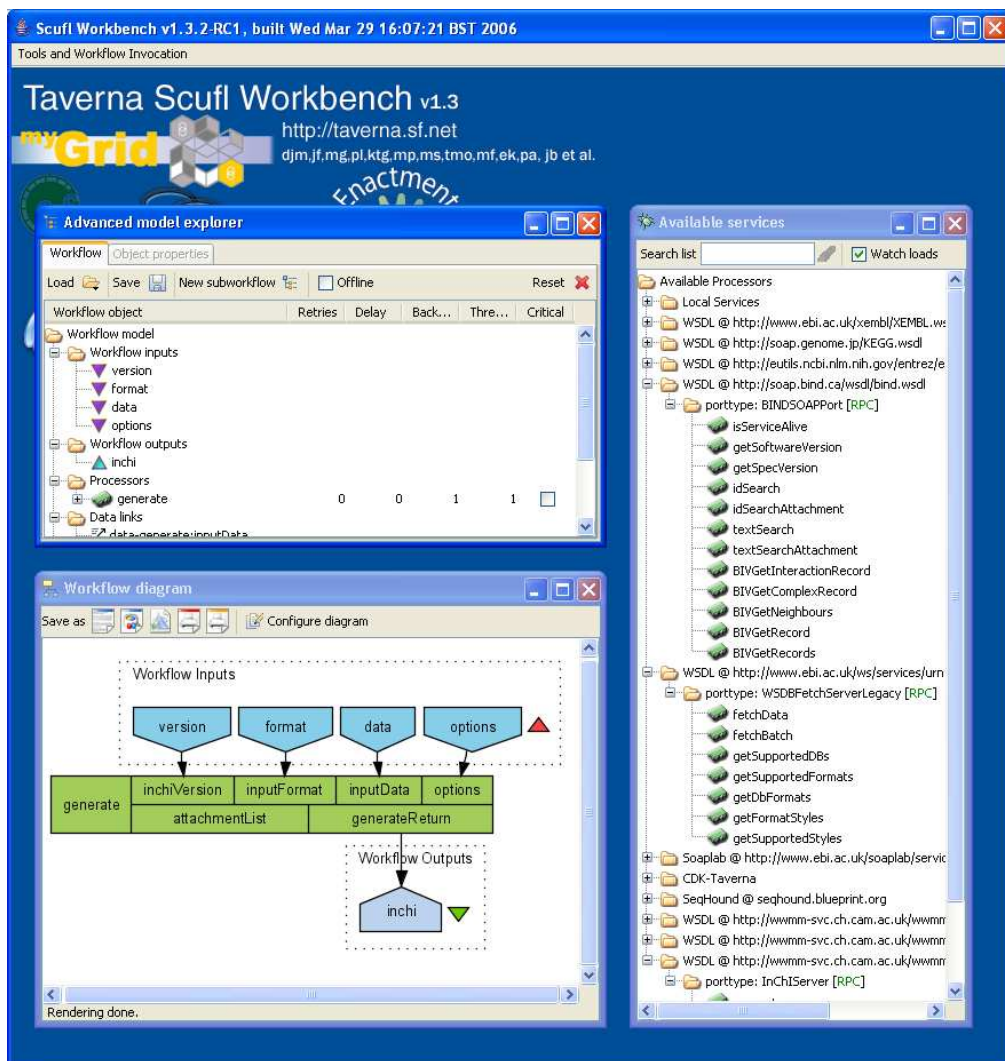


Figure 4.2: The Taverna GUI showing the advanced model explorer (top-left), the available services panel (right) and the workflow diagram (bottom-left). Note that almost all of the folders in the available services panel correspond to sets of WSs provided by members of the bioinformatics community (those folders with names that contain a URL).

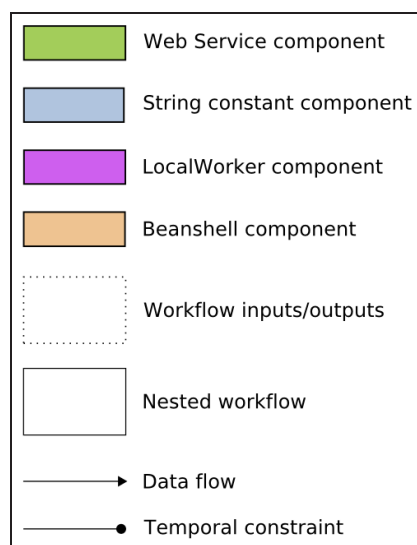


Figure 4.3: Key for the different workflow components, as rendered in the Taverna GUI.

#### 4.2.2 Component development and workflows

In order to provide the required functionality for the system, a set of LocalWorkers was created. Many of these provide wrappers to our pre-existing tools (*e.g.* JUMBO, CIFXML-J), though some also provide access to parts of 3rd party libraries (*e.g.* the XPath capabilities in XOM). Some of these components are shown in figure 4.5.

Along with the LocalWorkers, a set of SOAP-based WSs were developed and made available on the WWMM web server[139]. These were primarily to provide access to 3rd party command-line applications, such as InChI and OpenBabel (see figure 4.6). The workflow diagram window in figure 4.2 shows a workflow consisting solely of an InChI Web Service[141] component.

Using these components, two workflows were created for

1. finding and aggregating CIFs from journals by the IUCr and RSC (see figure 4.7),
2. the conversion of CIFs to enhanced CML (see figure 4.8).

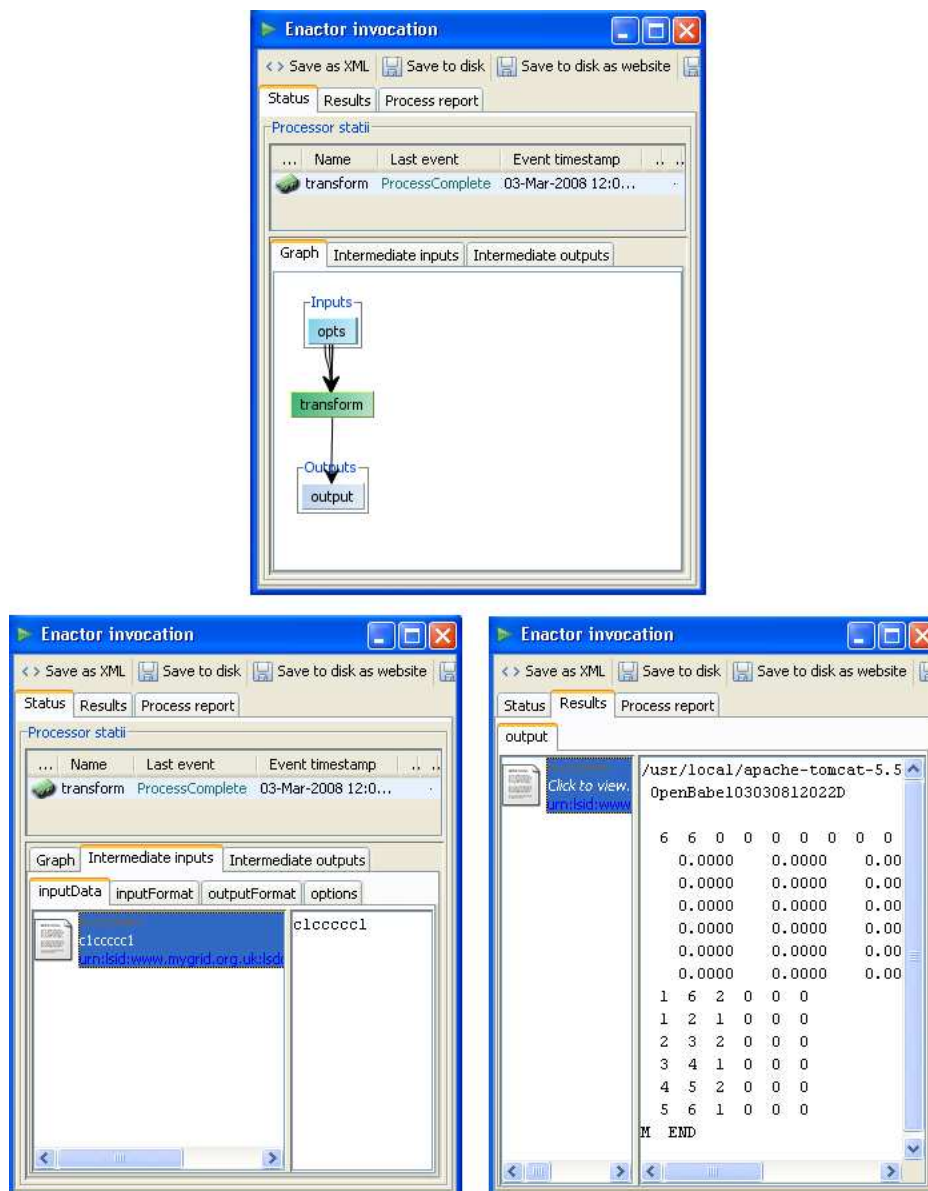


Figure 4.4: Three examples of the visualization provided by the enactor invocation window. The workflow involved contains one component, which is our WWMM OpenBabel WS[140]. The action being taken is to convert the SMILES representation of benzene into a MOL file. The images show the workflow progress (top), the data being passed to the 'inputData' input node of the WS (bottom-left) and the output of the workflow (bottom-right).

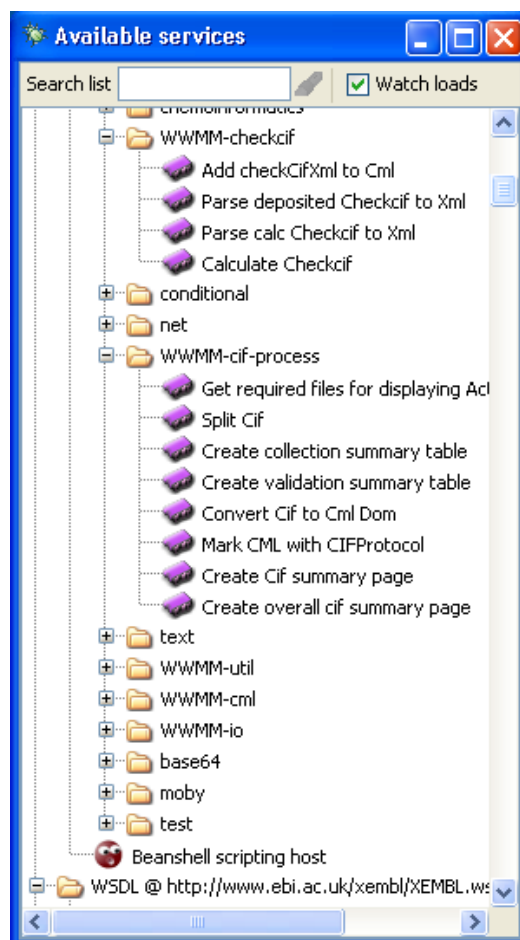


Figure 4.5: The available services panel showing the local services known to Taverna. Each folder prefixed with WWMM contains a category of our LocalWorkers.



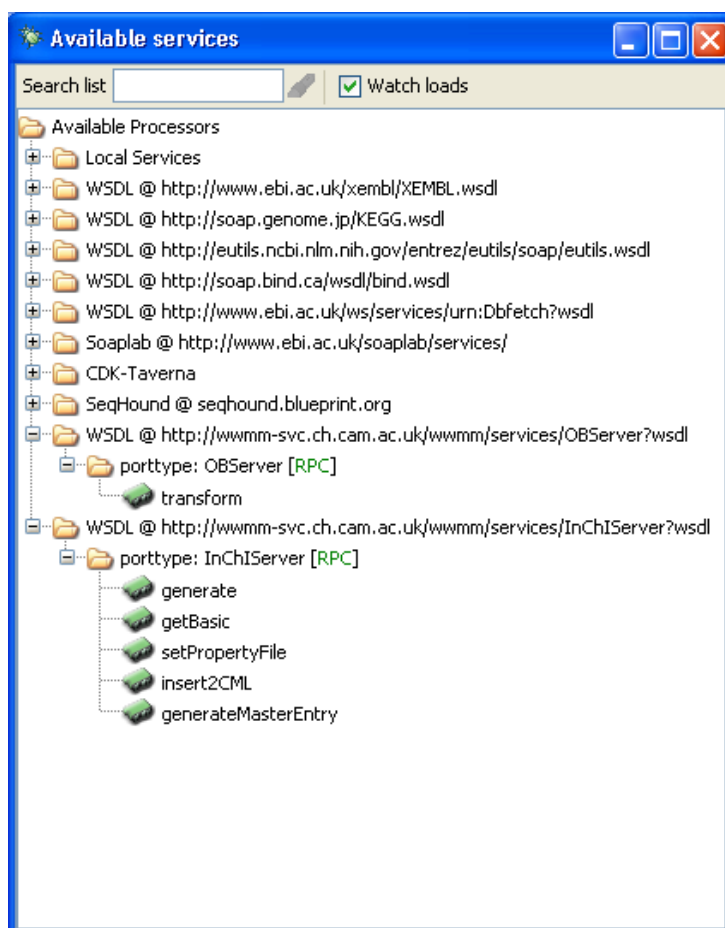


Figure 4.6: The available services panel showing the integration of the InChI and OpenBabel WSs (the last two folders). Note that the folder names show the URLs of the WSDL files for each WS.

As already mentioned, details of the processes involved in these workflows will be discussed in detail in section 4.3.

### 4.2.3 Developing under Taverna

Two of the primary goals of Taverna are to enable:

1. scientists with little or no computer background to create complex workflows by linking components provided by others through the GUI.
2. the ability not only to use distributed tools, but also distributed computer time, so that scientists without local access to powerful computers can access it remotely via WSs.

As one of the ideals behind the work in this thesis is creating Open Source software that can be easily reused by others, Taverna appeared to be a good foundation from which to start. Working under Taverna offers no benefits in terms of efficiency or speed of development for a developer of components, though it does provide a good method of enabling others to reuse those components and workflows that are created. Indeed, our set of LocalWorker components were shared with researchers at Indiana University, and work built on both these and our WSs have been described by both Pierce[142] and Kim[143].

However, while creating the workflows, a number of problems with developing under Taverna became apparent to the author. When combined, these caused a significant delay to the development cycle and compromised the reliability of the software. Eventually it was decided that the benefits of being able to develop software quickly and reliably outweighed the benefits of sharing the software easily, and thus the use of Taverna was discontinued. Four of the problems are listed below:

- At the time of the development, there was no way of running the workflows outside the GUI (despite requests to the Taverna mailing lists). As the system being developed was meant to be automatic, I needed to

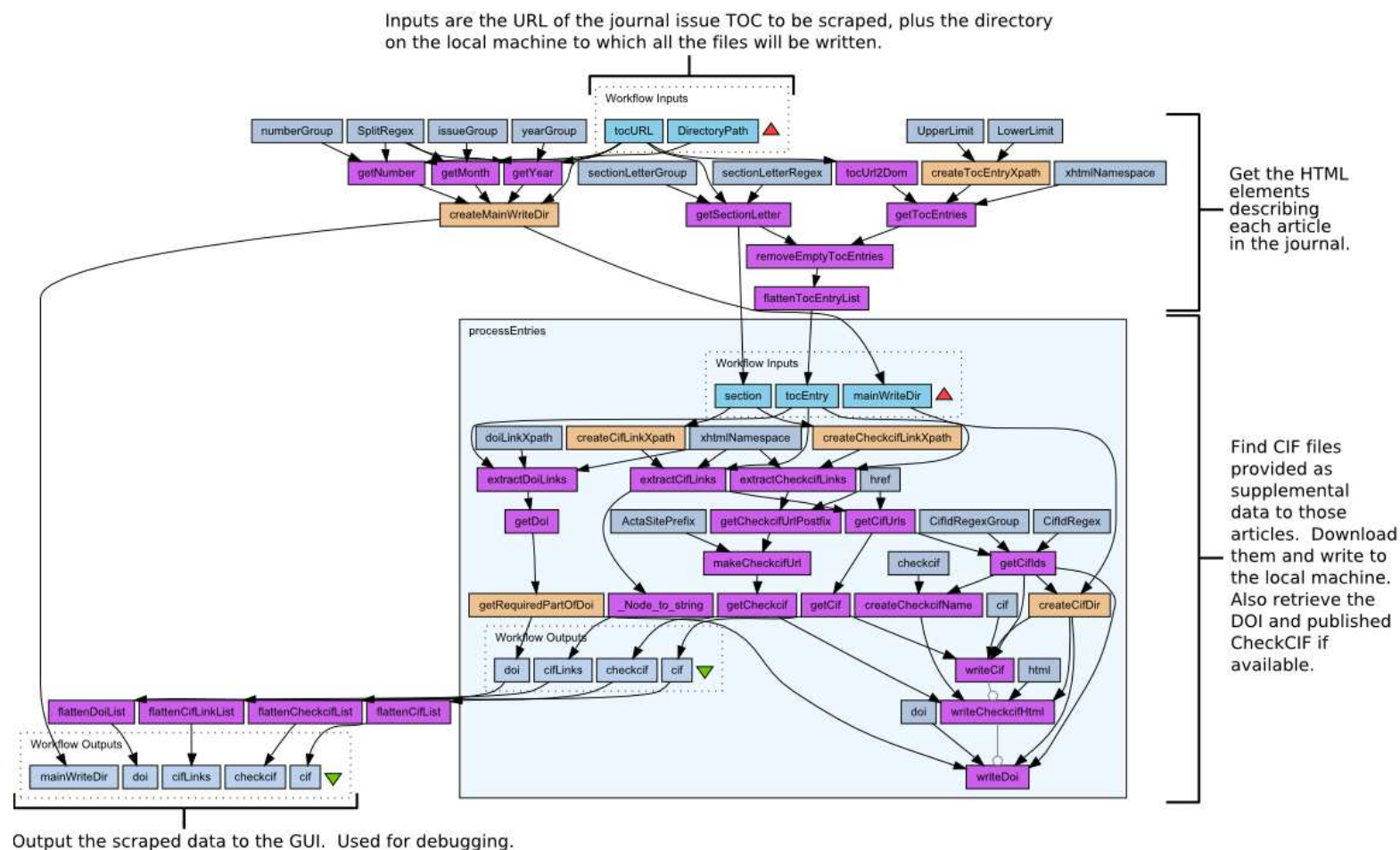


Figure 4.7: Annotated diagram (as rendered in the Taverna GUI) of the workflow used to find and aggregate CIFs and related data from IUCr journals

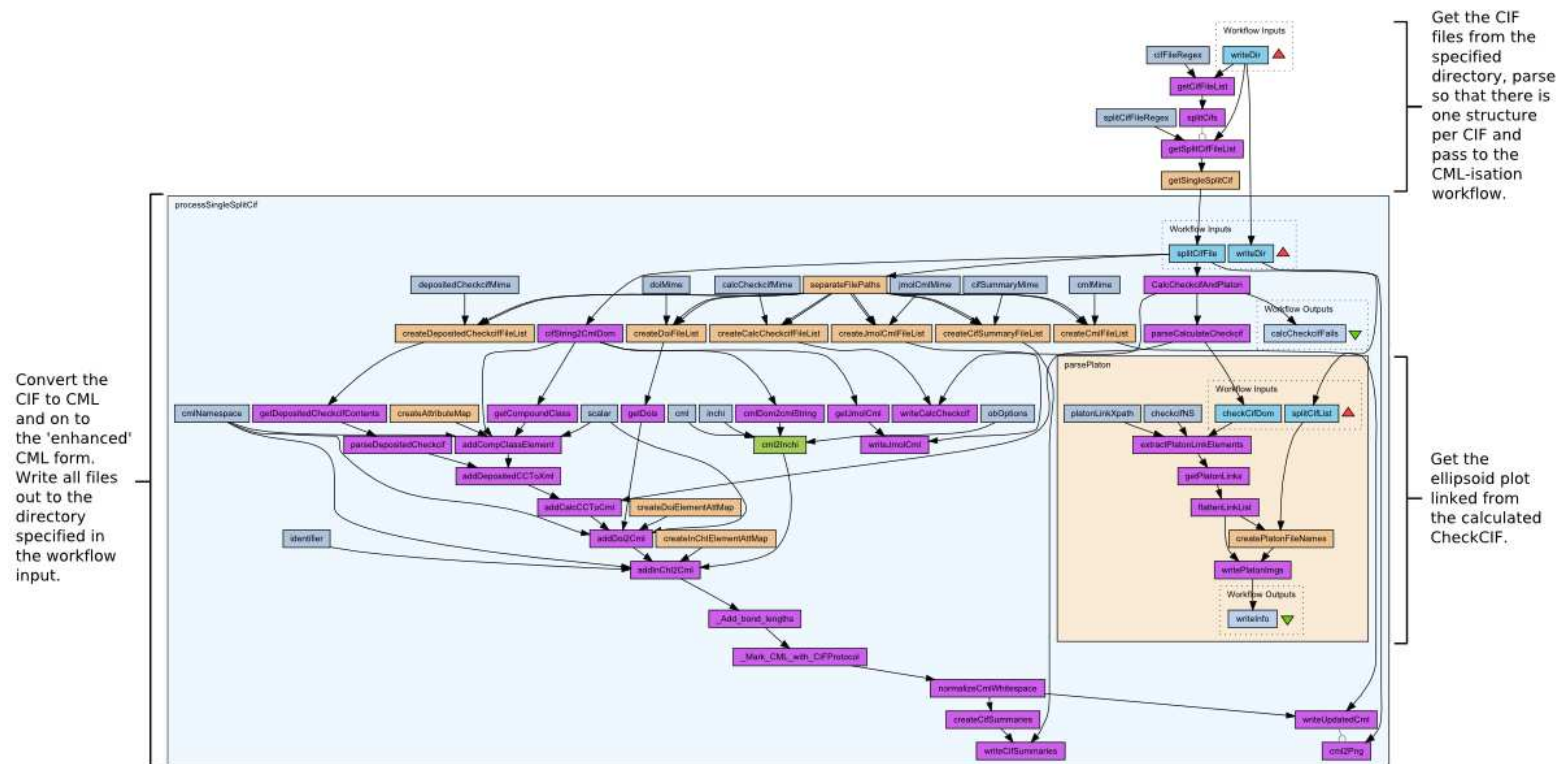


Figure 4.8: Annotated diagram (as rendered in the Taverna GUI) of the workflow used to convert CIFs to enhanced CML

provide a way for our web server to run the workflows periodically. The workflow enactment engine in Taverna is based upon **Freefluo**[144], an Open Source project based at Sourceforge (which had already been discontinued). I created a command line tool to run the workflows using the Sourceforge **Freefluo** library, though as it was unclear how similar the version of **Freefluo** in Taverna was, this was not a robust or predictable solution.

- When workflows are being run, a number of threads are created relating to the number of processors being run concurrently. As there was no way to limit how many threads were created, this led to large numbers being created for our complex workflows. This, coupled with the fact that large amounts of data were being processed meant that out-of-memory errors were common.
- When using nested workflows, the components contained within are not represented in the enactor invocation window (see figure 4.9), only the nested workflow itself. This makes debugging them difficult and time consuming, as it is not possible to directly see where the failure occurred, or the data that was being passed around to cause the failure. In order to debug, the nested workflow must be run on its own, meaning that the data causing the failure must first be captured and then passed in as a workflow input.
- It was not possible to test parts of a workflow in isolation. In order to do so, a new workflow consisting of only those components would have to be created (this was often faster than just running the entire workflow during each debug step). Thus, once a bug was found, the time taken to fix it, then check the workflow worked was greatly increased.

## 4.3 CrystalEye

After development of the Taverna-based system stopped, I refactored the code and created CrystalEye, which represents the reformulation of the frag-

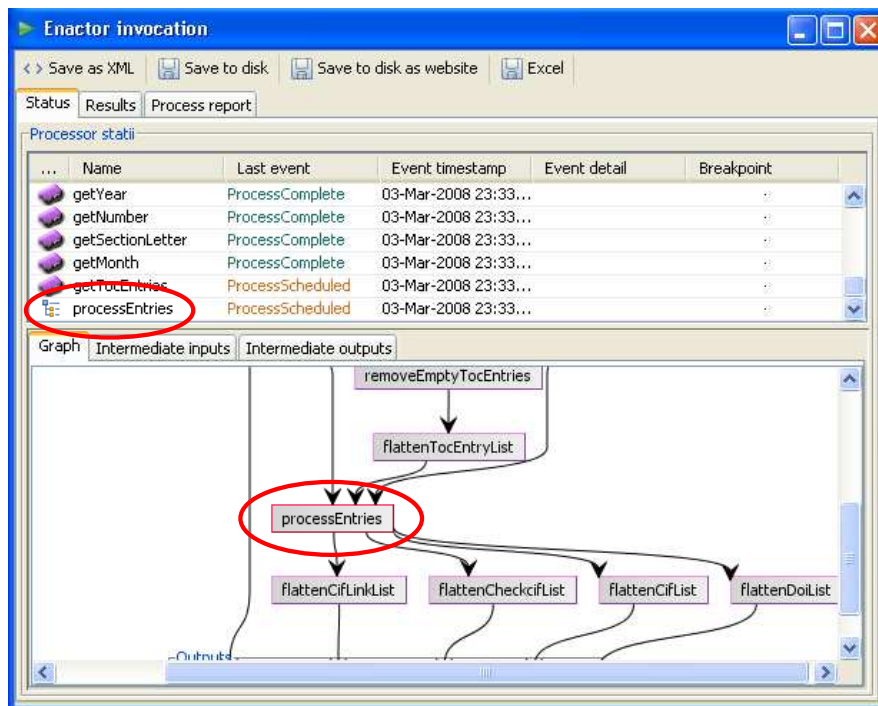


Figure 4.9: The enactor invocation window showing the workflow described in figure 4.7 being run. Notice that the large nested workflow titled ‘processEntries’ is represented as one component (circled), and that the details of the components within are not shown.

mented crystallographic web into a structured XML-based repository. CrystalEye runs continuously, performing the following tasks:

1. aggregating the latest Openly published CIFs from the Web,
2. converting the CIF data into CML,
3. updating a website containing services for dissemination of all aggregated and derived data.

All of the data aggregated and created by CrystalEye is Open, and a large part of the system is focussed on making the semantic content available using the latest Web technologies. By providing the data in this way, it is hoped that CrystalEye will promote and aid:

- the location of crystallography using Web search methods,
- the reuse of the crystallographic data in e-Science.

#### 4.3.1 Implementation

CrystalEye is a lightweight system which consists of a single standalone executable JAR (Java ARchive) file that uses a hierarchical file-system for data storage. This simple architecture is beneficial for distribution of the system as only Java need be installed to deploy it, and as the file-system is used for storage, all content can be made available in a RESTful[146] fashion (*i.e.* every resource has an address) by providing the containing directory on a web server (as is shown in section 4.3.4).

#### 4.3.2 Aggregation

To locate and aggregate newly published CIFs, a set of web-spiders that perform *web-scraping*[147] have been written. The resources which CrystalEye has aggregated CIFs from (as of 2008-07-03), are:

- Publisher's websites
  - Acta Crystallographica

- \* Section A: Foundations of Crystallography
- \* Section B: Structural Science
- \* Section C: Crystal Structure Communications
- \* Section D: Biological Crystallography
- \* Section E: Structure Reports
- \* Section J: Applied Crystallography
- \* Section S: Synchrotron Radiation
- The Royal Society of Chemistry
  - \* Chemical Communications
  - \* CrystEngComm
  - \* PCCP
  - \* Dalton Transactions
  - \* Green Chemistry
  - \* Journal of Materials Chemistry
  - \* New Journal of Chemistry
  - \* Organic and Biomolecular Chemistry
- The American Chemical Society
  - \* Analytical Chemistry
  - \* Bioconjugate Chemistry
  - \* Biochemistry
  - \* Biomacromolecules
  - \* Chemistry of Materials
  - \* Crystal Growth and Design
  - \* Inorganic Chemistry
  - \* Journal of Agricultural and Food Chemistry
  - \* Journal of the American Chemical Society
  - \* Journal of Combinatorial Chemistry
  - \* Journal of Chemical Information and Modelling
  - \* Journal of Medicinal Chemistry



- \* Journal of Natural Products
- \* The Journal of Organic Chemistry
- \* Langmuir
- \* Macromolecules
- \* Molecular Pharmaceutics
- \* Organic Letters
- \* Organic Process Research and Development
- \* Organometallics
- Elsevier
  - \* Polyhedron
- The Chemical Society of Japan
  - \* Chemistry Letters
- Open databases
  - COD (Crystallography Open Database)

All CIFs from the discontinued RSC journals, Perkin Transactions 1 and 2 have also been aggregated.

None of the above sites provide a formal method of discovering newly published CIFs, so I have developed methods for up-to-date aggregation for both categories of resource.

### **Aggregating from publisher’s websites**

The web spiders have been written around three libraries:

- `HTTPClient`[115] – provides methods for webpage retrieval, timeouts and retries. It is important to take the latter two points into consideration when writing spiders, as there are many reasons a URL request could fail initially, and this would lead to missed content.
- `TagSoup`[116] – used to tidy the retrieved HTML into markup that is valid XML.

- **XOM**[30] – reads the tidied HTML into an in-memory DOM representation, which can then be queried using XPath to find the desired data items.

Aggregation from each publisher is currently performed on a per journal issue basis. Each time the spiders are run, the initial step is to check the latest issue of each journal to see if it has already downloaded the CIFs from it. For most publishers, each journal has a "current-issue" URL that always points to the table of contents (TOC) of the latest issue to have been published. The URL below, for instance, always points to the latest issue of Dalton Transactions.

```
http://pubs.rsc.org/Publishing/Journals/dt/Article.asp?Type=
CurrentIssue
```

This can be edited to apply to any RSC journal by changing `dt` to the abbreviation of another of their journals. Once the TOC has been downloaded and read into **XOM**, it is queried using a predefined XPath to extract the year and issue number of the issue. The spider then deduces whether it needs to scrape that issue by comparing this data to an index of all previous issues to have been scraped.

The page hierarchy and HTML structure for journals of different publishers are generally not uniform (figure 4.10), but are the same for journals of the same publisher; thus, there is one spider for each publisher. The steps the spider performs when aggregating CIFs from a journal issue are highlighted in figure 4.11. The spiders for different publishers all apply a similar strategy, but differ in the number and structure of the XPath expressions that are needed to navigate through the webpages from the TOC to the CIFs. An extra step is required for Elsevier, as supplementary information for their articles is provided in a ZIP file. In this case, the spider must locate the ZIP file, unzip it and then check for CIFs. Note that for all journals, whenever a CIF is found, the spider will also extract the corresponding DOI for that article and store it as a file with the CIF.

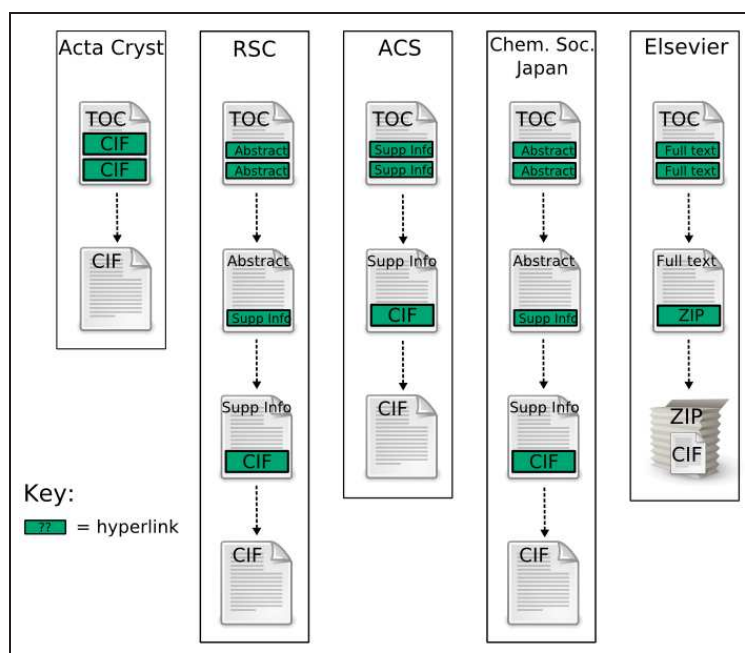


Figure 4.10: The page hierarchy that links the TOCs to the CIFs for each publisher scraped by CrystalEye.

In many cases, an article will not have an attached supplemental CIF, and so for some journals, the majority of the webpages downloaded *en route* to the supplemental pages are done so unnecessarily. Two other problems with this method of aggregation are that the spiders rely on:

1. *a fixed website structure.* If a publisher were to redesign their website, there is a chance that the XPaths would stop working. When writing the XPaths, I tried not to refer to the HTML structure, and only query for hyperlinks that contain HREFs with a particular folder structure (as it is less likely for the website folder hierarchy to be changed than the webpage HTML). Despite this, during the course of this work, the spiders for ACS and RSC journals both had to be rewritten as a result of website redesigns.
2. *the path from TOC to supplemental data being complete.* Publisher's websites are generated from fixed templates and so the spiders can rely

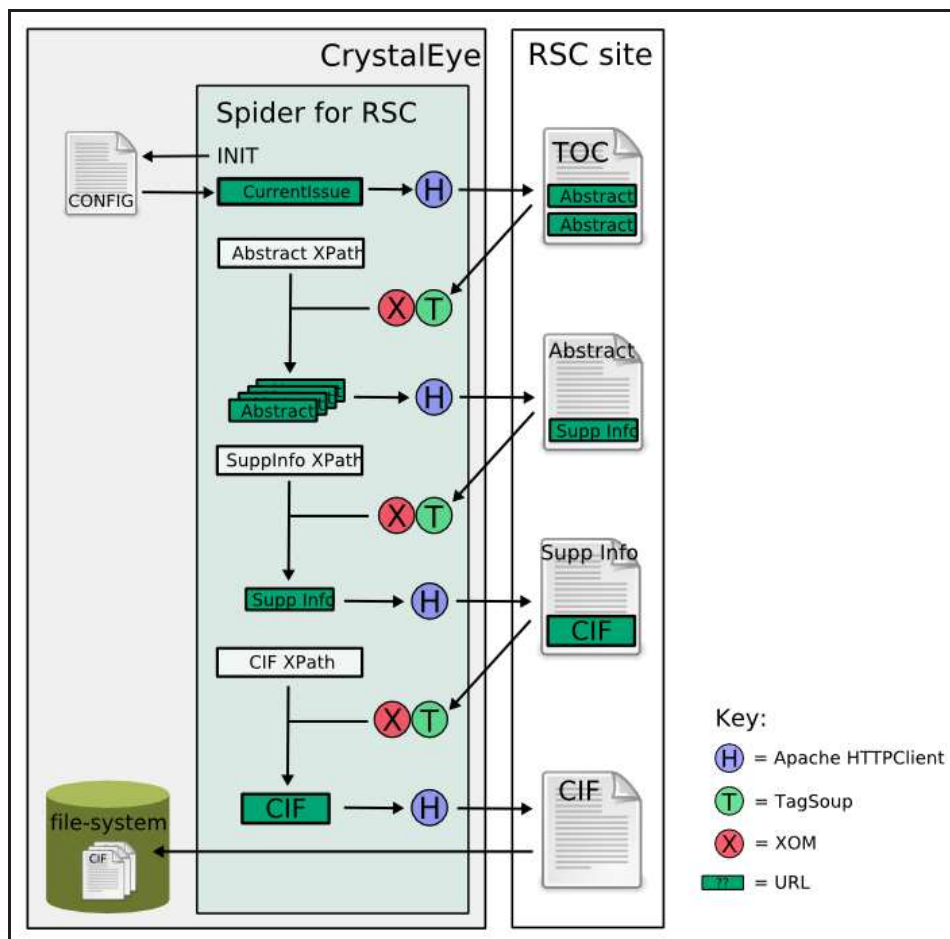


Figure 4.11: On initialization, the web-spider is provided with the ‘current-issue’ URL of the journal, which is then downloaded via the `HTTPClient` library. The webpage is tidied with `TagSoup`, read into a DOM using `XOM` and then queried with a predefined XPath to find all the links in the page to article abstract pages. The page referenced by each of these abstract links is then downloaded, tidied and queried to find any links to supplementary information pages. For each link the page is again downloaded and queried to find any links to CIFs. If any are found, they are downloaded and stored in the CrystalEye file system.

on a common HTML structure (barring website redesigns) to navigate to the supplemental data. However, humans may still neglect to enter required data into the template. As the spiders must navigate a website to find the desired content, if any of the expected hyperlinks are not provided along the way, then even if the final content is available, it will not be discovered. An example of this was noted when the spider was not returning any CIFs from a number of RSC journals in the latter half of 2004 (*e.g.* issues 14–23 of Dalton Transactions). On inspection, none of the abstract pages for articles in these issues contained links to supplementary data pages (which the spider relies on). To check whether supplemental data had been provided, but just not linked to, it is possible to use the journal code and article code to create the supplementary data URL that would have been provided on the abstract page (though by using this method, URLs for pages that don't exist may be created). The common form for these URLs is:

<http://pubs.rsc.org/suppdata/CE/b4/b416493h/index.sht>

where 'CE' is the journal code and 'b416493h' is the article code. A spider was written to extract the article codes from the TOC for each issue and then to create a supplementary data URL as above. An attempt was made to download the page at each URL, and many of them were found to exist and provide links to CIFs.<sup>§</sup>

Publishers now generally provide RSS feeds for their journals, where new feed entries are added for each article as they are made available on the site. In the near future, I am hoping to alter CrystalEye's aggregation, so that instead of the spider waiting for a new issue to be promoted to the 'current-issue' at a site, it could just read the RSS feed for that journal and follow each entry to the corresponding article. One benefit of this method would be that CrystalEye could get updated content as it was made available on the publisher's site, rather than having to wait until an entire issue had been finalised before scraping the whole thing.

---

<sup>§</sup>I notified the RSC of this problem, though it has yet to be rectified (2008-04-23).

These RSS feeds are still not a formal interface to the CIFs, and require some HTML spidering. At present the entries of these RSS feeds point to the abstract pages for an article, so CrystalEye would just be able to start its spidering process a little further along the chain. Thus, the problems described above must still be considered. Ideally the entries in these RSS feeds would provide links to any supplemental data files (*e.g.* using *enclosures*, see section 4.3.5). If this was used, no HTML scraping would be needed, and only one fixed-structure RSS document would have to be tracked by the spider in the search for new content. This would negate all of the previously described problems encountered during aggregation from publisher sites.

### **Aggregating from the COD**

The COD consists of CIFs that have been donated by publishers, institutions and researchers. As of 2007-11-14, the COD contains over 45,000 CIFs, and in particular provides a good source of inorganic structures, many of which have been donated by American Mineralogist. The database is available as a single large ZIP file[149] (> 120Mb, which contains the CIFs and a host of related information), hence only one URL is needed by the spider, which downloads the database, unzips it and checks through all the files for new CIFs. The COD is only updated a few times a year, so to save unnecessary downloading and processing, the spider can identify if the database has been updated by providing an 'If-Modified-Since'[150] header in the HTTP request. If it receives a 304 status (Not Modified) in return, then nothing more needs to be performed.

### **Dealing with duplicate CIFs**

Many of the crystal structures in the COD are duplicates of ones that CrystalEye has obtained from publisher's websites. In these cases, the version from the COD is of lower quality (often a pre-publication version), and so is discarded. To perform this data deduplication automatically, CrystalEye maintains an index of cell parameters against the chemical formula for each crystal that it aggregates.

Each time the COD is downloaded, every CIF in the database must be compared against this list before it is stored in CrystalEye. It is also possible that CrystalEye could obtain a CIF from the COD before it finds the same one on a publisher's website. For this reason, the system must also check the structures from each new journal issue download against all those it has previously obtained from the COD. The COD version will be removed if any duplicates are found.

### **Statistics for CIF aggregation**

The number of CIFs aggregated from each of CrystalEye's sources (as of the end of 2007) are:

- Acta Crystallographica – 30,465
- Royal Society of Chemistry – 10,351
- American Chemical Society – 26,548
- Elsevier – 168
- Chemical Society of Japan – 150
- Crystallography Open Database – 18,283

which gives the total number of CIFs aggregated as 85,965. Both Elsevier and the Chemical Society of Japan only provide CIFs in one journal each (as far as is known) and these only go as far back as 2006 and 2005 respectively. The growth in the number of CIFs aggregated per year can be seen in figures 4.12 and 4.13. The latter plot shows the recent rapid rise in the number of CIFs published in both ACS and Acta Cryst. journals, while the number in RSC journals has stayed largely the same.

To test that the spiders are aggregating all CIFs correctly, two issues were selected at random for each year from 2001-2007 for every journal aggregated by CrystalEye. These issues were then manually checked for supplementary CIFs and, for each, the number of CIFs found by hand matched those found by the spider.

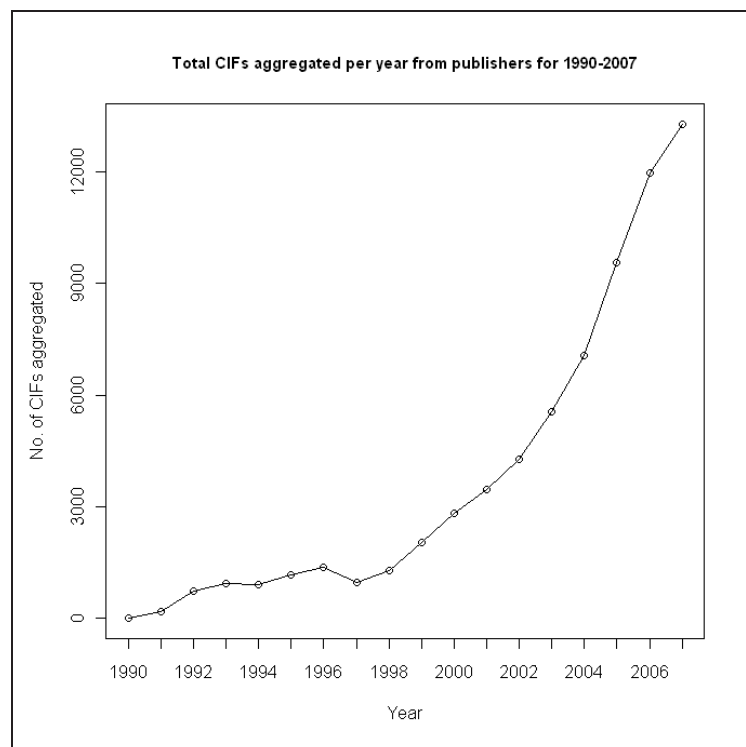


Figure 4.12: Plot showing the total number of CIFs aggregated from publisher's websites each year from 1990 to 2007.



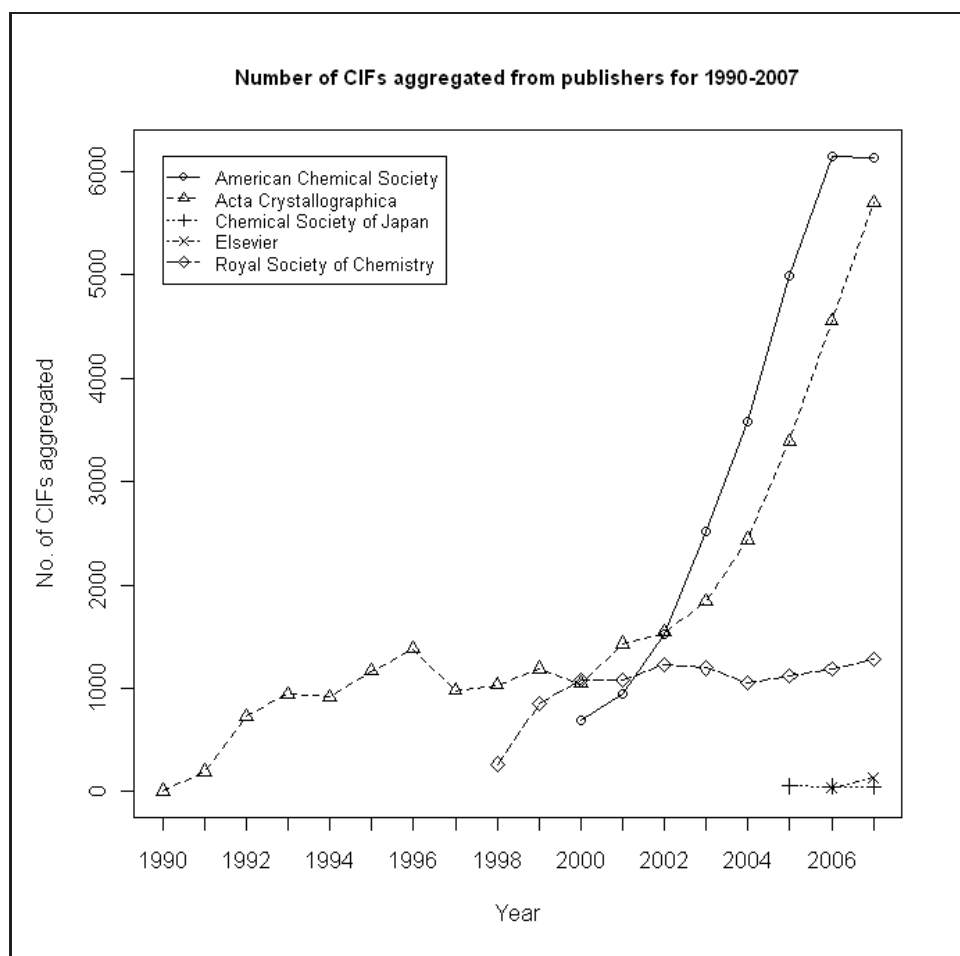


Figure 4.13: Plot showing the number of CIFs aggregated from each publisher from 1990 to 2007. Note the anomalous drop in CIFs published in ACS journals in 2007.

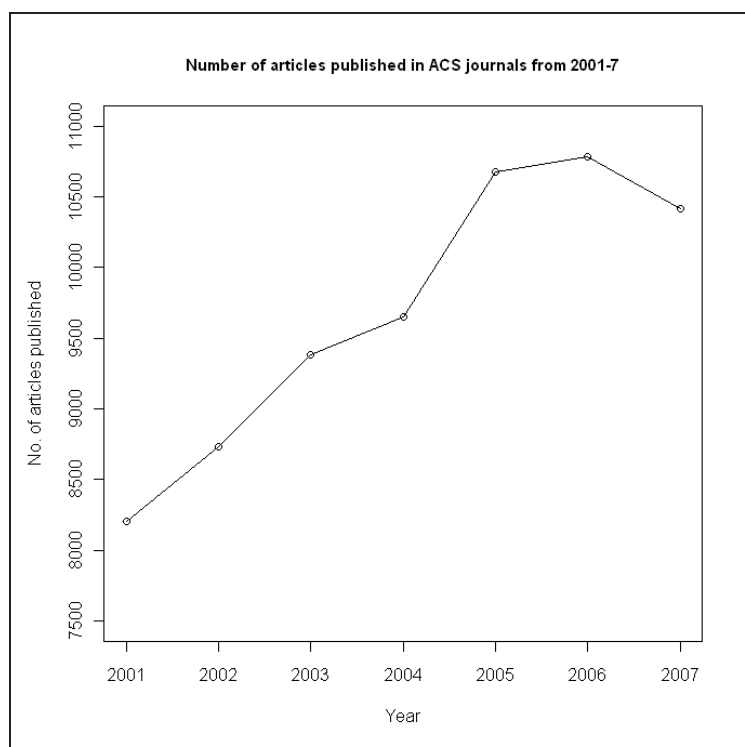


Figure 4.14: Number of articles published in ACS journals from which over 500 CIFs have been aggregated between 2001-7

If this is the case, then what is the cause of the anomalous drop in the number of CIFs aggregated from ACS journals in 2007? This may be partly explained by inspecting the number of articles published in those ACS journals for which over 500 CIFs have been aggregated since 2001 (figure 4.14). It can be seen that, like the number of CIFs aggregated, the number of articles increased from 2001 to 2006, but decreased in 2007.

If the number of CIFs published *per article* for these journals is viewed (figure 4.15), then we see that the number does rise each year, though the steady growth seen from 2001-6 is not continued from 2006-7. It is difficult to know what has caused the break in growth, though it may be that the limit for CIFs being provided for all relevant articles is being reached.

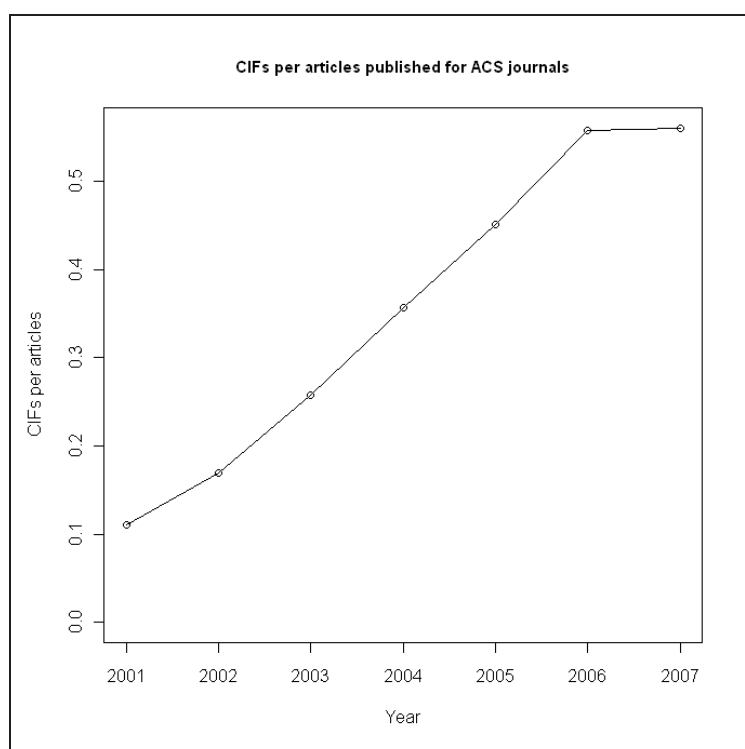


Figure 4.15: The number of CIFs provided per article in ACS journals from which over 500 CIFs have been aggregated between 2001-7.

### 4.3.3 Processing the data

Once the CIFs have been downloaded, they are converted into CML and other related files which are used to represent and disseminate the structures via the CrystalEye website (section 4.3.4). The unit of currency in CrystalEye is the structure, not the CIF, and so the process of 'splitting' the CIFs into single structures, as described in section 3.5.1, is used. As CrystalEye is based on a file-system, all data created is stored as separate text files.

#### Creating checkCIF XML

checkCIF has been designed to be used by humans via a web browser. The webpage (figure 3.4 in section 3.3.4) contains a HTML form through which a user can upload a CIF and submit it for checking on the IUCr's web-server. Once the checkCIF process has finished, the user's browser will be redirected to a webpage containing the checkCIF report (either in PDF or HTML depending on the user's selection). In order to programmatically call the checkCIF service, I have written a Java program which mimics this behaviour using a library that implements the HTTP protocol.

The interface to checkCIF on the homepage is a HTML form, which, when viewed in the source code of <http://checkcif.iucr.org/>, looks like:

```
<form method="post" enctype="multipart/form-data"
  action="http://dynhost1.iucr.org/cgi-bin/checkcif.pl">
  File name:
  <br>
  <input type="file" name="file" size="35">
  <input type="hidden" name="runtype" checked="checked" value="symonly">
  <input type="hidden" name="referer" checked="checked" value="checkcif_server">
  <br>
  <input type="submit" name="UPLOAD" value="Send CIF for checking">
  <p>Select form of checkCIF report</p>
  <input type="radio" name="outputtype" value="HTML" checked="checked">HTML
  <input type="radio" name="outputtype" value="PDF">PDF
</form>
```

The URL that the CIFs are sent to for processing is shown (<http://dynhost1.iucr.org/cgi-bin/checkcif.pl>) along with the options that need setting (the input elements). Using Apache HTTPClient[115] I wrote a wrapper program that creates a HTTP POST method to submit a CIF and the required

options to the above URL, after which it retrieves the resulting checkCIF report as a HTML page.

As discussed previously, the checkCIF service was intended to be used by humans. The checkCIF HTML structure has therefore been designed to be understandable by humans viewing it through a web browser, and as a result the underlying HTML contains no semantics. In order to extract the data from the presentation-centric HTML, I have also written a checkCIF parser.

Despite the HTML containing no semantics, as the documents are programmatically generated we can be confident that the same HTML element hierarchies and text patterns will occur from report to report. Thus we can create XPath expressions (albeit semantically void ones) to access the desired data and use it to create a new semantically rich checkCIF XML document.

To do this, the parser first tidies the HTML into valid XML using the `Tagsoup`[116] library, after which the tidied HTML can be parsed by the `XOM` library and the resulting DOM can be queried using XPath. The parser has a set of hardcoded XPath expressions which are run in sequence to access the DOM nodes which contain the desired data items. For each node retrieved, the parser will extract the child text and use it to build the new XML document. Figure 4.16 shows an example of the HTML for a set of alerts and the corresponding representation of those alerts in checkCIF XML. Note that checkCIF XML is not part of the CML Schema and therefore has a separate namespace.

An example XPath expression used by the parser is:

```
//x:a[contains(@href,'javascript:makeHelpWindow')]
```

While this query is intended to retrieve all of the HTML elements describing the alerts, the XPath itself bears no relation to the data whose location it describes. This is, however, the best way to uniquely identify the location of that data from the HTML structure provided. These XPaths are extremely

(a)

●Alert level C				
<a href="#">PLAT125 ALERT 4 C</a>	No _symmetry_space_group_name_Hall Given .....			?
<a href="#">PLAT230 ALERT 2 C</a>	Hirshfeld Test Diff for C11 - C13 ..		5.56 su	
<a href="#">PLAT242 ALERT 2 C</a>	Check Low Ueq as Compared to Neighbors for		C7	
<a href="#">PLAT242 ALERT 2 C</a>	Check Low Ueq as Compared to Neighbors for		C11	

(b)

```
<html>
...

<hr>

<FONT size="+1" color="#999900"><B>Alert level C</B></FONT>
<a href='javascript:makeHelpWindow("PLAT125.html")'>PLAT125_ALERT_4_C</a>
  No _symmetry_space_group_name_Hall Given ..... ?
<a href='javascript:makeHelpWindow("PLAT230.html")'>PLAT230_ALERT_2_C</a>
  Hirshfeld Test Diff for C11 - C13 .. 5.56 su
<a href='javascript:makeHelpWindow("PLAT242.html")'>PLAT242_ALERT_2_C</a>
  Check Low Ueq as Compared to Neighbors for C7
<a href='javascript:makeHelpWindow("PLAT242.html")'>PLAT242_ALERT_2_C</a>
  Check Low Ueq as Compared to Neighbors for C11
<hr>
...

</html>
```

(c)

```
<checkcif xmlns="http://journals.iucr.org/services/cif">
  <calculated>
    <dataBlock id="I">
      <alerts>
        <alert code="PLAT125_ALERT_4_C">
          <alertText>
            No _symmetry_space_group_name_Hall Given ..... ?
          </alertText>
        </alert>
        <alert code="PLAT230_ALERT_2_C">
          <alertText>
            Hirshfeld Test Diff for C11 - C13 .. 5.56 su
          </alertText>
        </alert>
        <alert code="PLAT242_ALERT_2_C">
          <alertText>
            Check Low Ueq as Compared to Neighbors for C7
          </alertText>
        </alert>
        <alert code="PLAT242_ALERT_2_C">
          <alertText>
            Check Low Ueq as Compared to Neighbors for C11
          </alertText>
        </alert>
      </alerts>
    </dataBlock>
  </calculated>
</checkcif>
```

Figure 4.16: Diagram showing the alerts section in a checkCIF report (a) when viewed through a web browser, (b) when viewed as the HTML source code and (c) after they have been converted to checkCIF XML.

fragile with respect to change in the checkCIF HTML (*e.g.* a change in the name of the Javascript method associated with the alert links would cause the above XPath expression to cease to work), and a more suitable method for extracting data from checkCIF reports will be needed in the future. This could either be provided by including semantically named attributes on the HTML elements to describe the contained data, or preferably, by providing optional XML output from the checkCIF service.

### Creating the CML

Each CIF is processed through the CIF to enhanced CML process described in chapter 3. After this, the following are also merged into the CML

- the DOI for the article from which the structure was scraped. In the CML, the DOI is described as a scalar, *e.g.*

```
<scalar dictRef="idf:doi">10.1107/S1600536808004169</scalar>
```

The DOIs are used widely on the CrystalEye website to provide permanent hyperlinks back to a structure's parent article (the URL for a DOI is created by prepending <http://dx.doi.org/>).

- the checkCIF XML. Thus the crystallographic data is provided alongside explicit community standard crystallographic validation. Reuse of the data is aided by the ability of users to make assertions on its quality.

For each structure aggregated by CrystalEye, the final CML file contains:

- the original CIF experimental and researcher metadata,
- enhanced CIF structural data, which is either the complete unit cell (for inorganic and polymeric structures), or the unit cell containing the complete unique moieties (for non-polymeric organic and organometallics). If possible, the latter are enhanced with:

- resolved disorder,

- bond orders and charges,
- 2D and 3D stereochemical information,
- 2D atomic coordinates,
- SMILES and InChI canonical identifiers,
- the structure’s DOI
- the checkCIF XML

## 2D structure diagrams

For each non-disordered organic and organometallic moiety, a 2D image is generated from the CML using the CDK (see figure 4.17). This works well for most organic structures, though for some organic and many organometallic structures the software, understandably, produces images where atoms and bonds overlap. This gives diagrams where it is difficult to see the actual structure, and in some cases, the structure appears to be wrong (see figure 4.18). The use of templates for common metal coordinations and ring-systems has been suggested as a part solution, though for a universal solution, other methods will need to be considered (further discussion in section 4.3.8).

## Generating molecular fragments

For each non-disordered organic and organometallic moiety, CML files are generated for the following molecular fragments (as shown in figure 4.19):

- ring-nuclei,
- metal ligands,
- ring-ring, ring-terminus and terminus-terminus chains,
- metal centres,
- metal clusters.

Fragments are also created for singly and doubly sprouted fragment nuclei (except for metal ligands, which are already complete), as shown in figure 4.20.



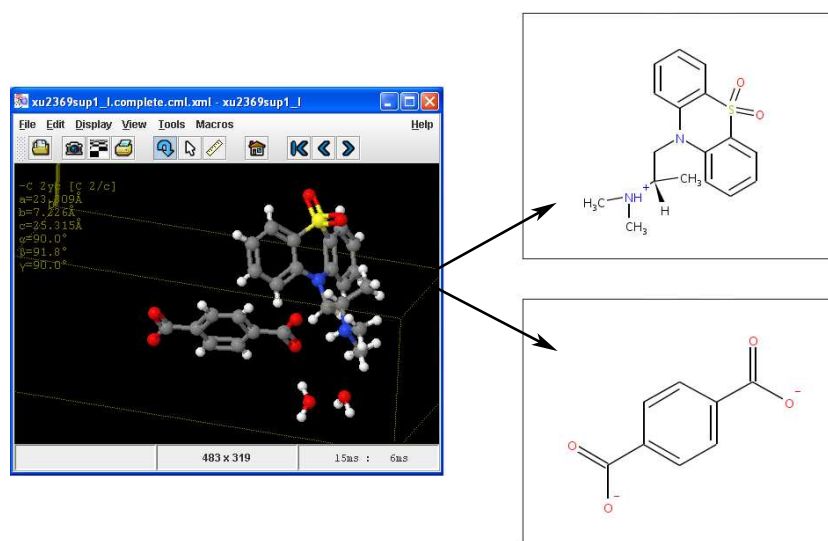


Figure 4.17: Rendering of the CML for *rac*-Bis[1-(9,9-dioxo-10*H*-phenothiazin-10-yl)-2-propyl] dimethylammonium terephthalate trihydrate[151] in Jmol with the 2D images (complete with stereochemistry) generated for the contained moieties.

#### 4.3.4 The website

A website[152] is maintained by CrystalEye to allow all of the generated content to be browsed. As the CIF aggregation is performed on a per-issue basis, each time a new issue is scraped, an interactive table of contents is generated and a new link is added to the issues that have been scraped for that journal (as in figure 4.21). The CrystalEye TOC allows all of the crystal structures from that issue to be browsed quickly in both 2D and 3D. Access is provided to further information about each structure and also links back to the original journal article using the DOI (see figure 4.22).

For each crystal structure, a summary page is created highlighting important metadata, and providing access to all of the related files (see figure 4.23). These pages are important as they are used by the services (section 4.3.5) as the reference point for each structure.

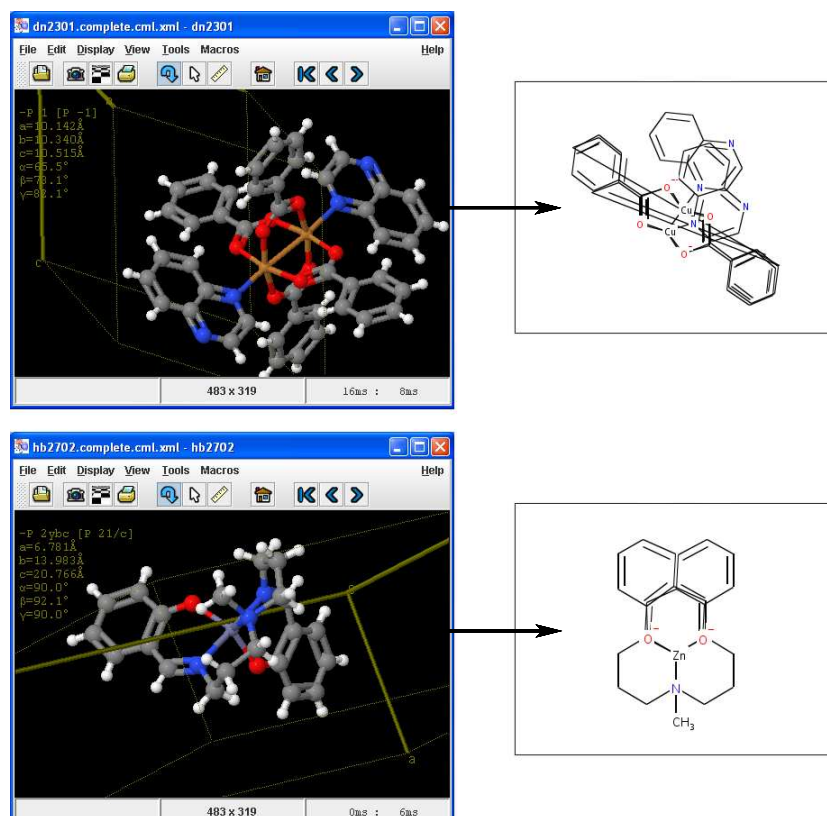


Figure 4.18: Rendering of the CML for (top) Tetra- $\mu$ -benzoato-bis[(quinoxaline)copper(II)] and (bottom) {2,2'-[4-Methyl-4-azaheptane-1,7-diylbis(nitrilomethyldiene)]diphenolato}zinc(II) in Jmol with the corresponding 2D images showing overlapping atoms and bonds.

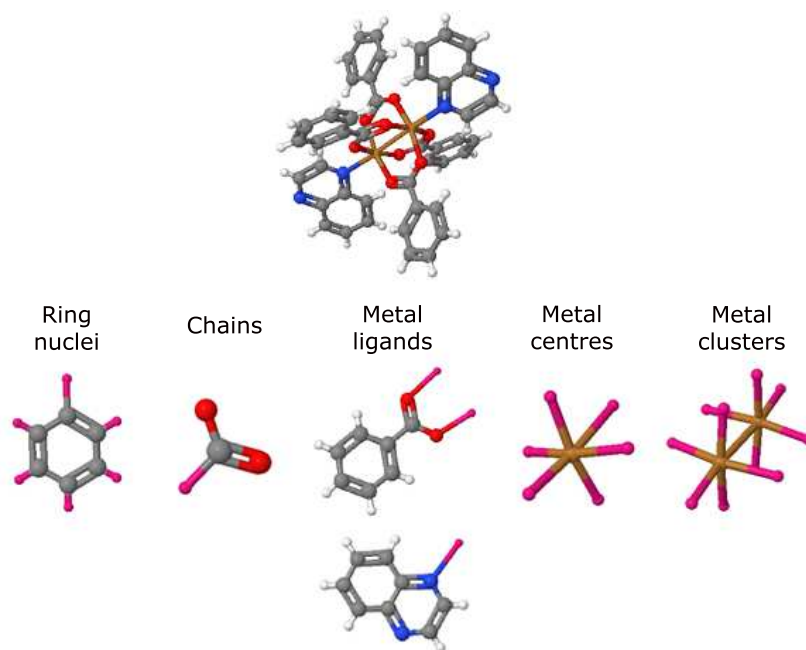


Figure 4.19: Example of the fragments generated from Tetra- $\mu$ -benzoato-bis[(quinoxaline)copper(II)]. The pink atoms indicate the fragment R groups.

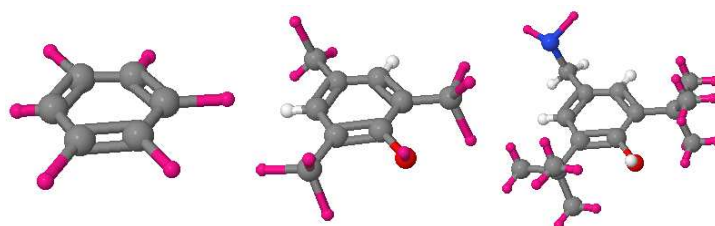


Figure 4.20: Example of a ring-nucleus fragment that has been singly and doubly sprouted.

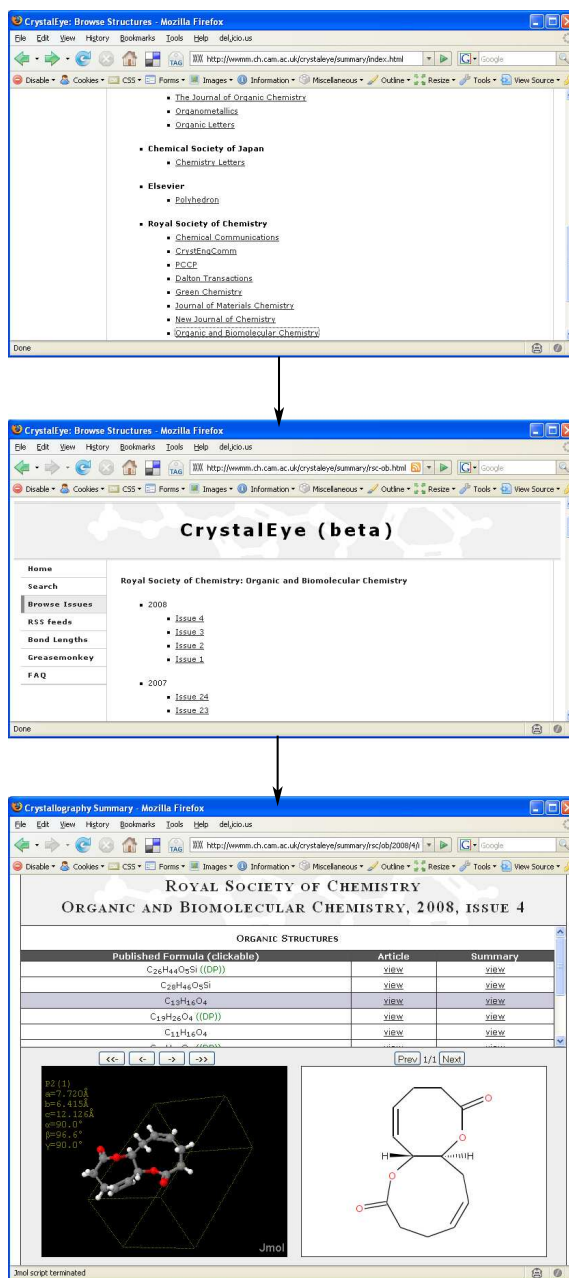


Figure 4.21: By following the links on the webpage summarising which journals CrystalEye aggregates CIFs from (top), the user can see all of the issues for which CrystalEye found crystal structures (middle). Clicking on an issue brings up a webpage which allows the user to browse all the crystal structures associated with that issue. Each row in the given table corresponds to one crystal structure. By clicking on a row, or by using the navigation buttons, the structure will be displayed in both 2D and 3D at the bottom of the page.

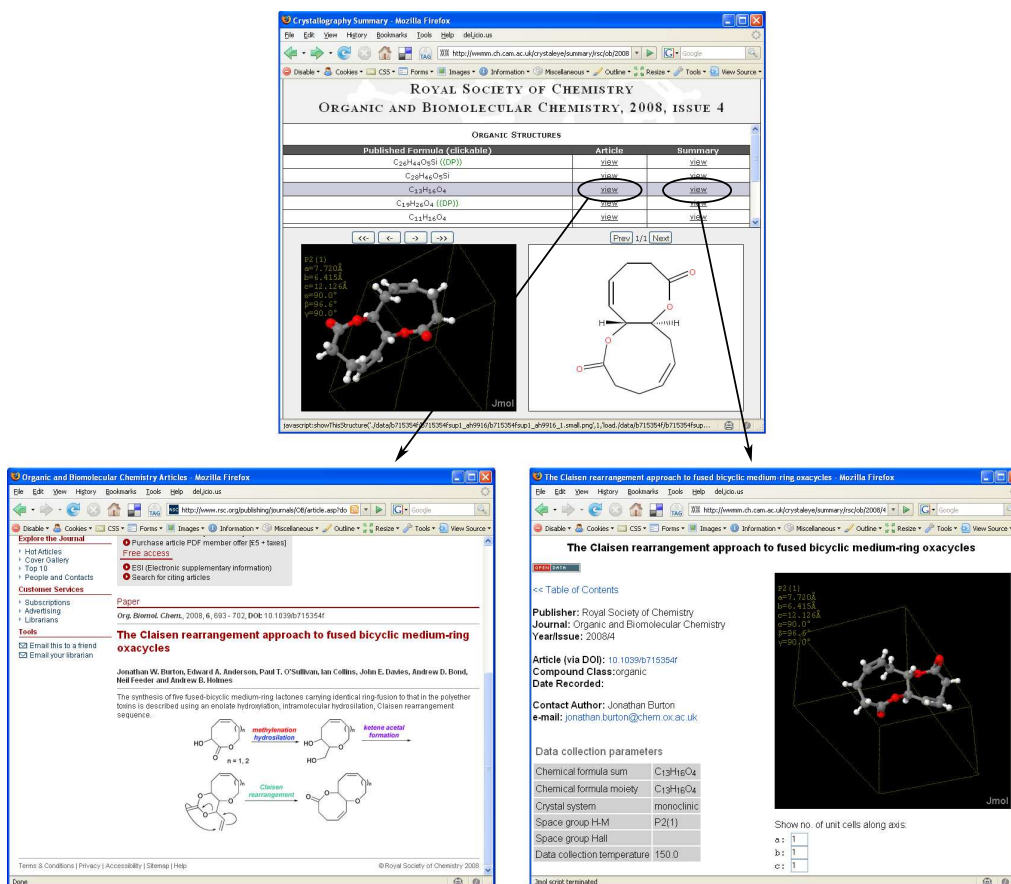


Figure 4.22: From the issue summary page, the user can access both the original article (via the DOI) or see the summary of all information CrystalEye has aggregated and generated for each structure.

**The Claisen rearrangement approach to fused bicyclic medium-ring oxacycles**

[Open Data](#)

[<< Table of Contents](#)

**Publisher:** Royal Society of Chemistry  
**Journal:** Organic and Biomolecular Chemistry  
**Year/Issue:** 2008/4

**Article (via DOI):** [10.1039/b715354f](https://doi.org/10.1039/b715354f)  
**Compound Class:** organic  
**Date Recorded:**

**Contact Author:** Jonathan Burton  
**e-mail:** [jonathan.burton@chem.ox.ac.uk](mailto:jonathan.burton@chem.ox.ac.uk)

**Data collection parameters**

Chemical formula sum	C <sub>13</sub> H <sub>16</sub> O <sub>4</sub>
Chemical formula moiety	C <sub>13</sub> H <sub>16</sub> O <sub>4</sub>
Crystal system	monoclinic
Space group H-M	P2(1)
Space group Hall	
Data collection temperature	150.0

**Refinement results**

R Factor (Obs)	0.0455
R Factor (All)	0.0603
Weighted R Factor (Obs)	0.1179
Weighted R Factor (All)	0.12

**Available Resources**

Crystal Components

[Moieties](#)

Result files

[Raw CML](#)

[Complete CML](#)

[CIF \(cached / original\)](#)

Validation

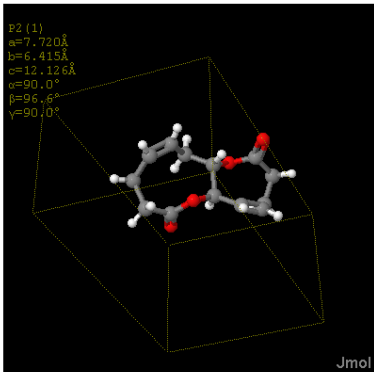
[CheckCIF](#)

Images

[Ellipsoid](#)

**InChI:** InChI=1/C13H16O4/c14-12-8-3-1-2-6-10-11(17-12)7-4-5-9-13(15)16-10/h1-2,4,7,10-11H,3,5-6,8-9H2/b2-1,-7-4-/t10,-11+m/s1

**SMILES:** [H]C1=C([H])C([H])([H])([C@]2(OC(=O)C([H])([H])C([H])([H])C([H])=C([H])(C@]2([H])(OC(=O)C([H])([H])C1([H])([H])([H]))([H])



Show no. of unit cells along axis:

a:

b:

c:

Enter Jmol script:

```
load./b715354fsup1_ah9911.complete.cml.xml
```

Figure 4.23: Image showing a full webpage of a crystal structure summary in CrystalEye. This includes: a symbol that it is Open Data (top-left), journal and contact author details and a link back to the original article (via the DOI), a summary of important crystallographic metadata, the 3D structure shown in Jmol (with the ability to enter arbitrary Jmol scripts), links to all of the generated files for that structure, the canonical identifiers for the crystal (to allow search engine indexing of the connection tables).

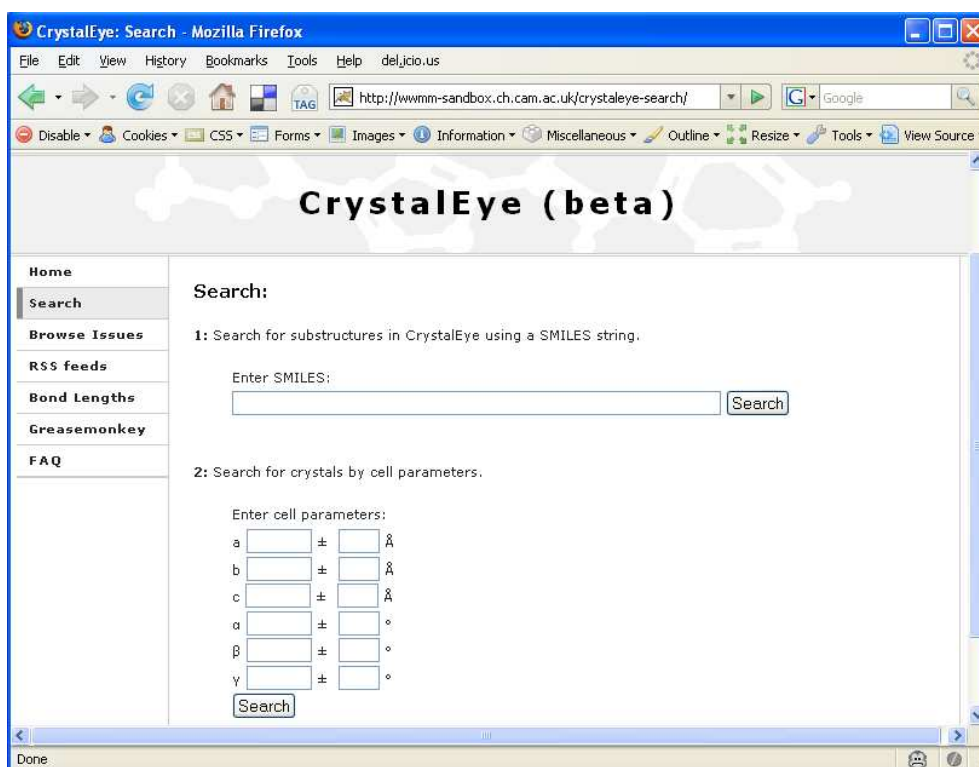


Figure 4.24: The CrystalEye search page

### 4.3.5 Services

Once the CML has been generated for all the newly aggregated CIFs, a set of services is then automatically updated with the latest data. As the data is stored in CML, accessing the data items required to update the service indexes is trivial.

#### Structure and cell search

There are currently two methods of searching the structures in CrystalEye (see figure 4.24):

1. Substructure search – OpenBabel is used for both creating an index of all the SMILES strings in CrystalEye, and for performing the searching of that index. An example of a substructure search is shown in figure 4.25).

2. Cell parameter search – allows the user to enter an arbitrary amount of cell parameter detail for the structures they want to find (see figure 4.26).

Other searches, such as exact structure matching with InChI/InChIKey, will be added when time permits.

## RSS and CMLRSS

In order to allow users to be notified when new structures of a particular type are published, CrystalEye maintains a large set of RSS and CMLRSS feeds. At present, these feeds are in the following categories:

- *Journal* – structures from a particular journal.
- *Compound class* – structures that are either organic, inorganic or organo-metallic.
- *Atom* – structures that contain a particular element.
- *Bond* – structures that contain a bond between two particular elements.

To provide for the most commonly used newsfeed types, each feed is created as RSS 1.0, RSS 2.0 and Atom 1.0 (see figure 4.27). A person who is interested in structures containing carbon-silicon bonds could subscribe to that feed[158] and they will be notified via their RSS reader each time such a structure is found (as in figure 4.28).

As shown in figure 4.28, each entry in the feeds contain a link to the structure webpage in CrystalEye. The only difference between the entries in the RSS and the CMLRSS feeds is that the latter have the complete CML document for the corresponding structure as a child element. Doing this allows CML aware RSS readers to automatically render each structure (*e.g.* the CMLRSS reader in Bioclipse[160]). The text below shows portions of the carbon-silicon bond Atom 1.0 CMLRSS feed, highlighting how the CML document for a given structure is merged directly into the corresponding feed entry.



1: Search for substructures in CrystalEye using a SMILES string.

Enter SMILES:

↓

## CrystalEye (beta)

**Home**

**Search**

**Browse Issues**

**RSS feeds**

**Bond Lengths**

**GreaseMonkey**

**FAQ**

**Searching...**

Search results for SMILES string "[Cu]n1cccc1" (in reverse chronological order) :

Showing results 1 to 25 of 1177

---

1: OC(=O)c1cc2[n](cc1)[Cu]1([O-]P2(=O)O)[n]2ccc(cc2P(=O1)([O-])O)C(=O)O  
[The American Chemical Society, Crystal Growth and Design, 2008, 3, article cq700673x, cif 2, datablock 70116am](#)

2: c1ccc(cc1)[B-]1(CP([Cu](P(C1)(C(C)(C)C)C(C)(C)C)[n]1cccc1)(C(C)(C)C)C(C)(C)C)c1cccc1  
[Royal Society of Chemistry, Chemical Communications, 2008, 9, article b713687k, datablock nrm08](#)

↓

**Diazoalkanes react with a bis(phosphino)borate copper(I) source to generate [Ph2BPtBu2]Cu(1-N2CR2), [Ph2BPtBu2]Cu(CPh2), and [Ph2BPtBu2]Cu?N(CPh2)(NCPH2)**

[OPEN DATA](#)

[<< Table of Contents](#)

**Publisher:** Royal Society of Chemistry  
**Journal:** Chemical Communications  
**Year/Issue:** 2008/9

**Article (via DOI):** [10.1039/b713687k](https://doi.org/10.1039/b713687k)  
**Compound Class:** organometallic  
**Date Recorded:**

**Contact Author:** Prof. Jonas Peters  
**e-mail:** [jcpeters@mit.edu](mailto:jcpeters@mit.edu)

**Data collection parameters**

Chemical formula sum	C <sub>35</sub> H <sub>55</sub> BCuNP <sub>2</sub>
Chemical formula moiety	
Crystal system	Orthorhombic
Space group H-M	Fdd2
Space group Hall	

The structure displayed is the major occupied structure from the crystal.

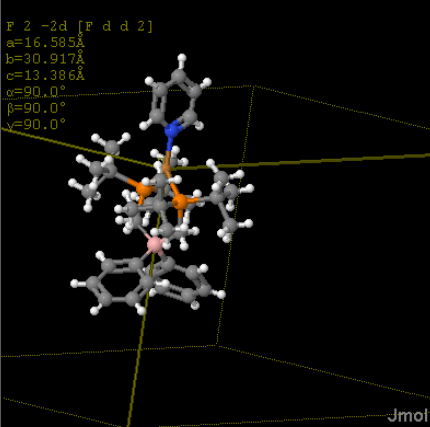


Figure 4.25: A substructure search of CrystalEye for all copper atoms bonded to the nitrogen atom of a pyridyl ring

2: Search for crystals by cell parameters.

Enter cell parameters:

a  ±  Å

b  ±  Å

c  ±  Å

α  ±  °

β  ±  °

γ  ±  °

↓

**Searching...**

Search results for cell parameters **a=9.0 (± 0.01)** , **b=14.0 (± 0.1)** , c=-, **α=90.0 (± 0.01)** , **β=-** , **γ=-** (returned in reverse chronological order) :

Showing results 1 to 2 of 2

---

1: a=9.004, b=14.001, c=15.273, α=90.0, β=100.16, γ=90.0  
[The American Chemical Society, Journal of the American Chemical Society, 2002, 42, article ja026704l, datablock 4-0-5toluene-0-5water](#)

2: a=9.0079, b=14.0545, c=18.9564, α=90.0, β=101.564, γ=90.0  
[Royal Society of Chemistry, Chemical Communications, 2005, 46, article b510081j, datablock 2\(2-3-4-pyr-nap\)-2\(fum\)-SCSC](#)

---

Figure 4.26: A cell parameter search of CrystalEye

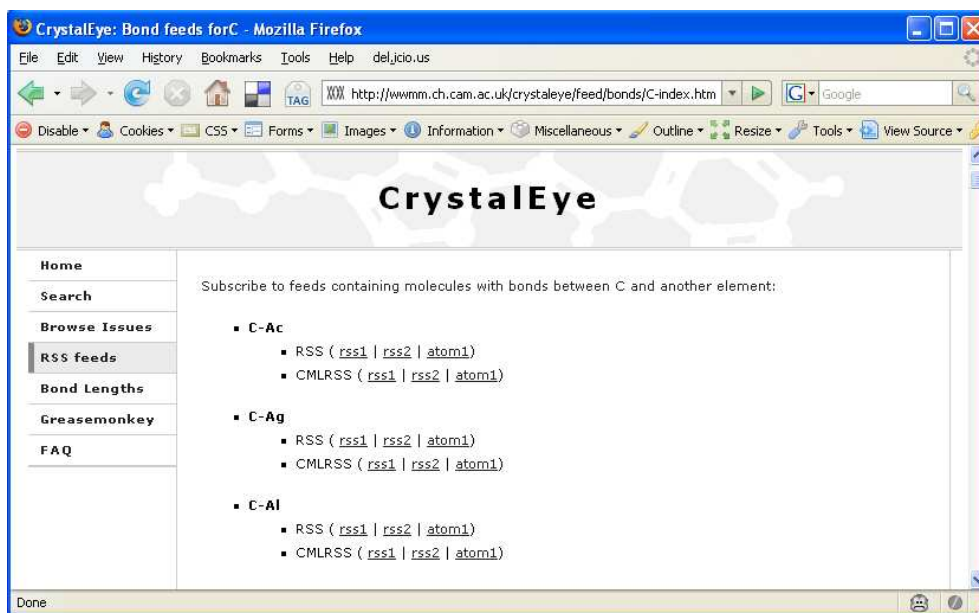


Figure 4.27: Portion of the webpage providing links to RSS and CMLRSS feeds of all structures contain a bond between carbon and another element

```
<feed xmlns="http://www.w3.org/2005/Atom"
  xmlns:cml="http://www.xml-cml.org/schema"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:sy="http://purl.org/rss/1.0/modules/syndication/"
  xmlns:taxo="http://purl.org/rss/1.0/modules/taxonomy/">
<title>
  CrystalEye CMLRSS: Structures containing bonds of C-Si
</title>
<link rel="self"
  href="http://wwwm.ch.cam.ac.uk/crystaleye/feed/bonds/C-Si/cmlrss/atom_10/feed.xml">
</link>
...
<entry>
  <title type="html">
    Catalyst- and solvent-free conditions as an environmentally benign approach to
    4-aryl-3-cyano-hexahydro-4H-1,2-benzoxazine-2-oxides
  </title>
  <summary type="text">
    CrystalEye CMLRSS summary of DataBlock 11a in CIF B712858D (DOI:10.1039/b712858d)
    from issue 3/2008 of Royal Society of Chemistry, Green Chemistry.
  </summary>
  <cml title="11a" id="rsc_gc_2008_3_b712858dsup1_11a" xmlns="http://www.xml-cml.org/schema">
    ...
    <scalar
      dictRef="iucr:_cell_measurement_temperature"
      dataType="xsd:double"
      errorValue="2.0">
        295.0
      </scalar>
```

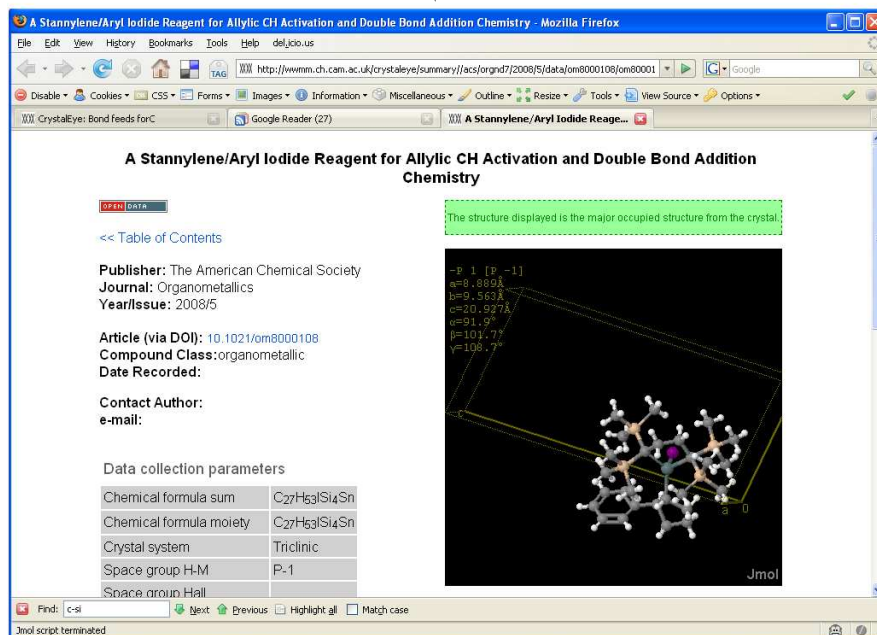
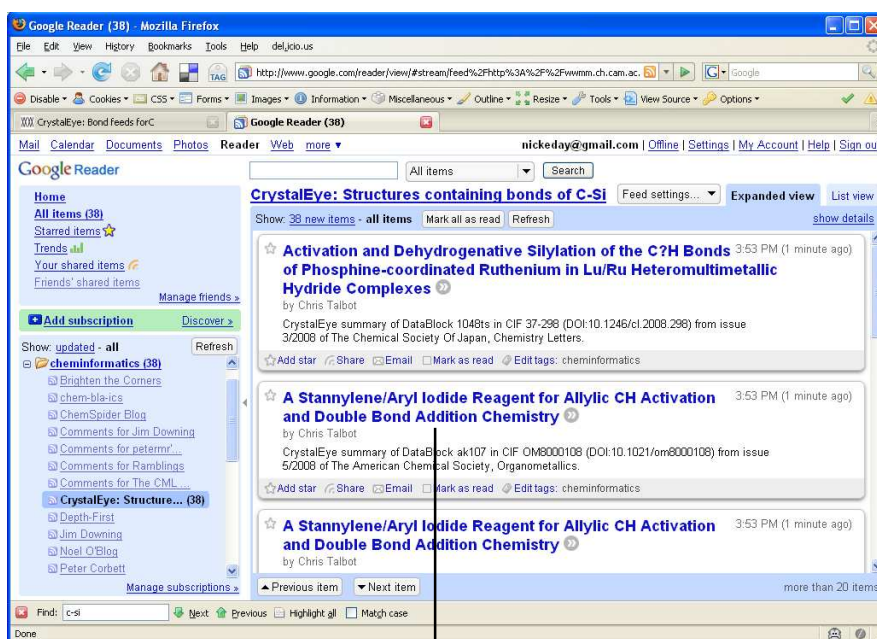


Figure 4.28: The RSS 2.0 feed for structures containing carbon-silicon bonds shown in Google Reader[159]. Clicking on an entry redirects the browser to the CrystalEye summary page of that structure.

```

    <scalar
      dictRef="iucr:_refine_ls_r_factor_all"
      dataType="xsd:double"
      errorValue="0.0">
        0.2485
    </scalar>
    ...
    <molecule id="rsc_gc_2008_3_b712858dsup1_11a">
      <crystal z="2">
        ...
      </crystal>
      <atomArray>
        ...
      </atomArray>
      ...
    </molecule>
  </cml>
</entry>
...
</feed>

```

The method of containing the CML explicitly in a feed is how CMLRSS was defined in the initial paper[161]. However, CMLRSS feeds with many entries, or with large amounts of CML in each entry can become prohibitively large. For instance, the journal feed for Acta Crystallographica Section E will receive many hundreds of new structures per issue, and this can lead to a feed that is tens or hundreds of megabytes large. Each time the feed is read by an RSS reader the whole document is downloaded before the reader decides which entries are new, and thus which ones to highlight for the user. With explicit CML in each entry there can be a large amount of CML downloaded for previously viewed entries, and thus a large percentage of the bandwidth used is unnecessary.

An additional problem is that for many readers the default method of parsing is to read the entire document into memory before processing. For these large documents this could cause problems such as out-of-memory errors. Indeed, in CrystalEye, to deal with these large feeds, the default method of parsing had to be changed from reading the entire document into a DOM using XOM to using a streaming XML parser (StAX[162]) to build sections of the DOM as-and-when needed.

An alternative method of providing the CML content is not to provide the markup explicitly in the feed document but to take advantage of *enclosures*, which are a feature of the Atom Syndication Format specification[163]. These allow content related to a feed entry to be provided as an enclosed link, which RSS readers can then follow to retrieve the content. This is not equivalent to explicit CMLRSS, as in this case only those CML documents for entries that have not previously been viewed will be retrieved. Enclosures allow an arbitrary number of files to be associated with an entry, so it is possible to include links to a number of files in CrystalEye that are related to each crystal structure. The CMLRSS feeds at CrystalEye still provide CML explicitly, though now all Atom 1.0 RSS feeds provide the content as enclosures. The text below shows portions of the carbon-silicon bond Atom 1.0 RSS feed.

```
<feed xmlns="http://www.w3.org/2005/Atom"
  xmlns:taxo="http://purl.org/rss/1.0/modules/taxonomy/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:sy="http://purl.org/rss/1.0/modules/syndication/"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
  <title>
    CrystalEye: Structures containing bonds of C-Si
  </title>
  <link rel="alternate"
    href="http://wwwm.ch.cam.ac.uk/crystaleye/feed/bonds/C-Si/rss/atom_10/feed.xml" />
  ...
  <entry>
    <title>
      Catalyst- and solvent-free conditions as an environmentally benign approach to
      4-aryl-3-cyano-hexahydro-4H-1,2-benzoxazine-2-oxides
    </title>
    <link rel="alternate"
      href="http://wwwm.ch.cam.ac.uk/crystaleye/summary//rsc/gc/2008/3/data/b712858d/
        b712858dsup1_11a/b712858dsup1_11a.cif.summary.html" />
    <link rel="enclosure"
      href="http://wwwm.ch.cam.ac.uk/crystaleye/summary//rsc/gc/2008/3/data/b712858d/
        b712858dsup1_11a/b712858dsup1_11a.complete.cml.xml" hreflang="en" />
    <link rel="self"
      href="http://wwwm.ch.cam.ac.uk/crystaleye/summary//rsc/gc/2008/3/data/b712858d/
        b712858dsup1_11a/b712858dsup1_11a.cif.summary.html" hreflang="en" />
    ...
  </entry>
  ...
</feed>
```

By providing all feeds as RSS and CMLRSS, and each in three different versions, there are almost 60,000 feeds currently maintained by CrystalEye. Implementing all possible feeds that a user may wish to subscribe to like this is not a scalable strategy and much processing time and storage space is

ultimately wasted. In the future I hope to implement an RSS system similar to that provided at del.icio.us[164], that is, the user specifies what they want to subscribe to, and the website dynamically creates that feed for them when new content is added. Using this method, it would be possible to provide RSS feeds by substructure by having the user enter their requirements as a SMILES string.

### Bond-length histograms

Indexes are maintained for the lengths of all bonds of each combination of elements in CrystalEye. These are converted to interactive histograms using the SVG graphing library created by Townsend[165]. A user viewing one of these histograms is able to click on a bin and browse all of the affiliated structures<sup>¶</sup> (see figure 4.29). Two histograms are provided for each element combination (see figure 4.30):

*All* : includes all bonds of that combination of elements, except those that contain atoms that are disordered or constrained (a fixed value placed on a parameter during crystallographic refinement),

*After protocol* : uses the same filters as in *All*, but the containing crystal structure must also have been refined at  $\leq 200\text{K}$ , and must have an R-factor (a measure of the agreement between the crystallographic model and the experimental X-ray diffraction data) of  $\leq 0.05$ .

### CrystalEye Greasemonkey

Greasemonkey[153] is an extension for the Mozilla Firefox[154] web browser that is able to alter the content of webpages after they have been downloaded. Users can create and install site-specific Greasemonkey scripts to add functionality to web pages (*e.g.* by combining data from another webpage via XMLHttpRequest[155]).

---

<sup>¶</sup>note that this interactivity is currently only available when viewing the histograms with the Mozilla Firefox browser

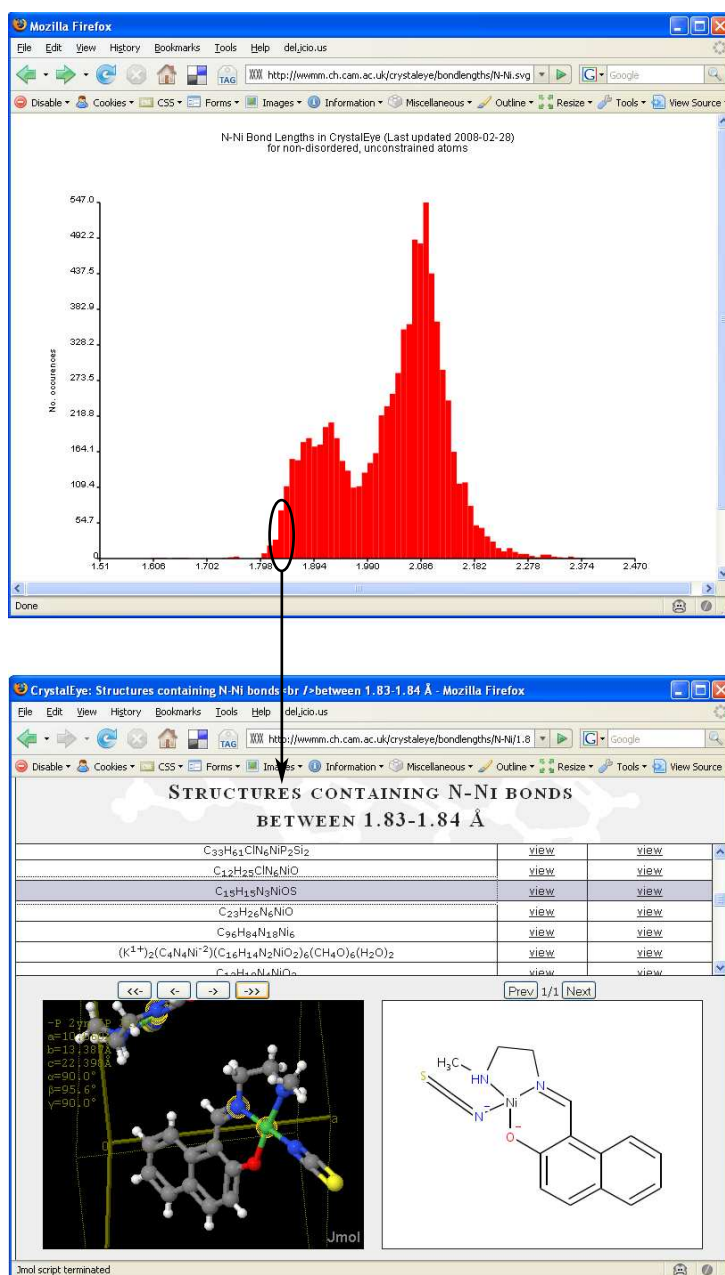


Figure 4.29: The top image shows the interactive histogram of the lengths of all nickel-nitrogen bonds in the structures in CrystalEye (as shown in the graph title, it was last updated by the system on 2008-02-28). Clicking on a bin brings up a webpage summarizing all those structures that constitute the bin. In the 3D rendering of the structure, the atoms in the corresponding bond are highlighted by a yellow halo.



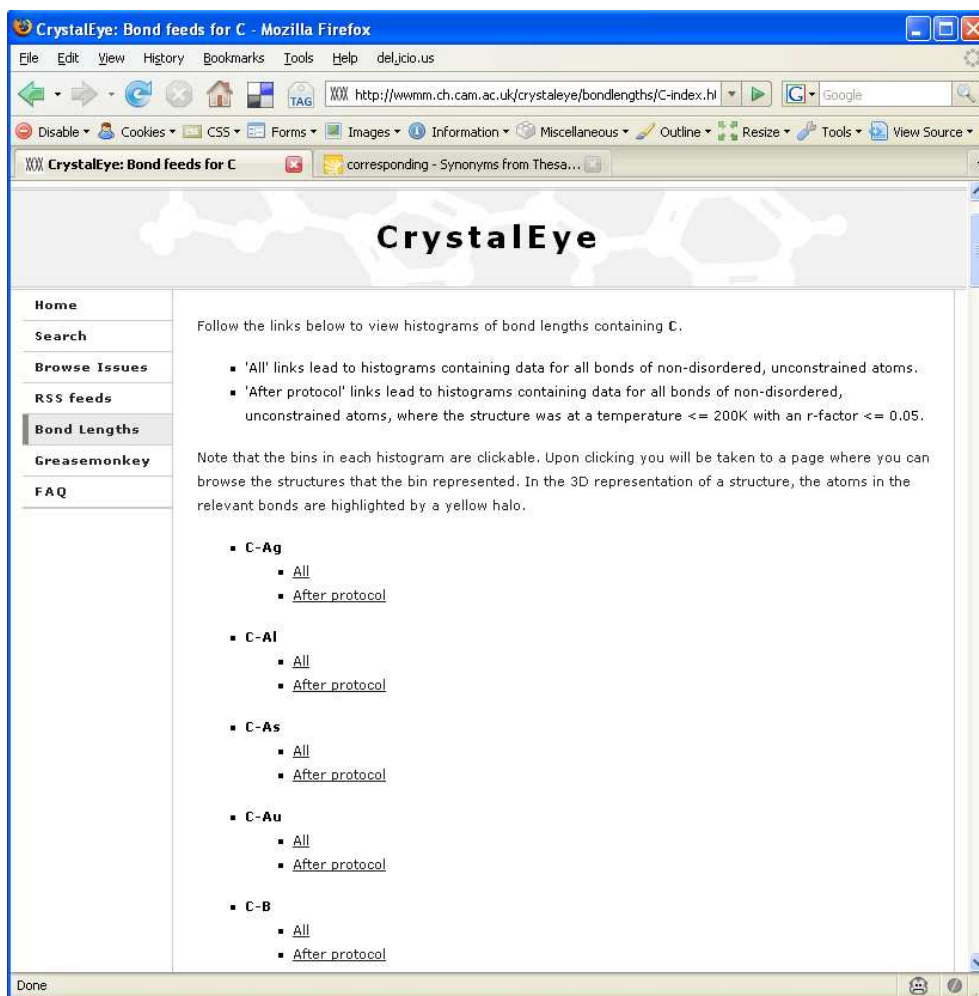


Figure 4.30: Portion of the webpage containing the list of links to the bond-length histograms for carbon

Many scripts have been described for the cheminformatics domain[156], and a CrystalEye Greasemonkey script[157] has been created, which targets webpages from any publisher scraped by CrystalEye. This site-specific targeting is achieved by providing a set of regular expressions in the script, for example

- <http://pubs.rsc.org/>\*
- <http://journals.iucr.org/>\*

which target all journal webpages at the RSC and IUCr sites. As previously discussed, the DOI is noted for each CIF aggregated by CrystalEye, which are then indexed against the URLs of the webpages of the corresponding crystal structure summaries. With the script installed, each time a user views a webpage that matches one of the script's regular expressions, the following actions automatically occur

1. the webpage HTML is searched for any DOI strings,
2. for each DOI found, it will be compared against the DOI index at the CrystalEye site,
3. if a match is found in the index, the CIF summary URLs at CrystalEye are noted (there may be more than one structure associated with a given DOI),
4. these URLs are then inserted as hyperlinks alongside the corresponding DOI in the HTML of the webpage being viewed.

With this script, a user is able to access all the functionality provided for the corresponding crystal structure by CrystalEye from the point of publication (figure 4.31 shows an example).

#### 4.3.6 Database distribution

Providing the ability to download and keep an up-to-date copy of the entire CrystalEye database is important for others who may want to use the data in their work, provide a mirror system or aggregate the data into a larger repository.

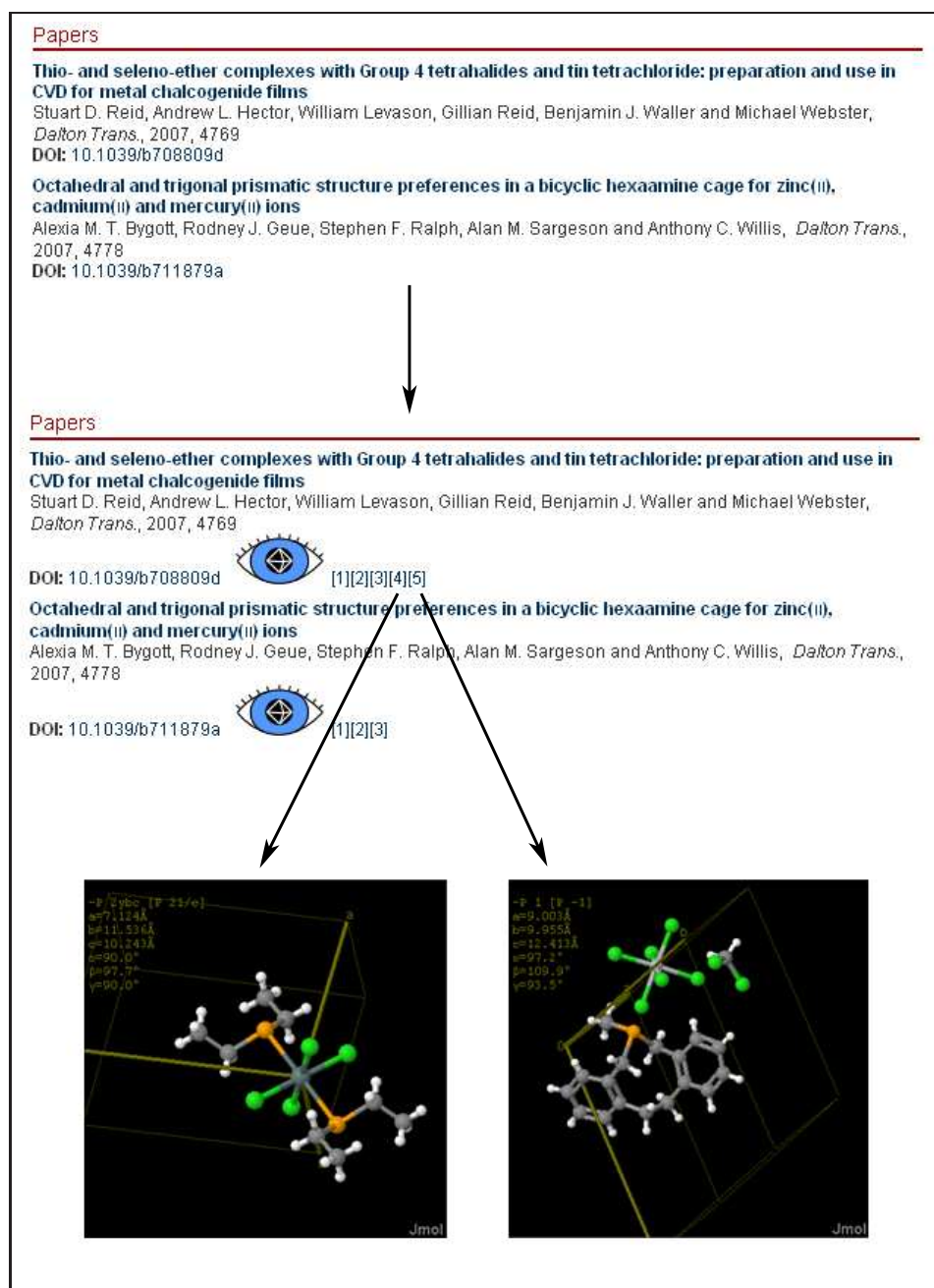


Figure 4.31: Example of the CrystalEye Greasemonkey script in action. The top image displays a portion of the TOC webpage for an issue of Dalton Transactions. Below is the same webpage viewed with the script installed, showing the hyperlinks and corresponding crystal structures at CrystalEye.

The total size of the CML for all structures in CrystalEye adds up to several gigabytes. As CrystalEye is updated on a daily basis, a mechanism must be provided to allow others to easily update their version of the data, without having to download previously obtained content (thus wasting bandwidth for both parties). Methods which were considered, but not implemented are discussed below. These would have been by providing:

- a monolithic ZIP file, which would be updated each time new structures were aggregated. However, by providing the data in this way, a user would have to download the entire database each time they wanted to update their version. For users who wanted daily updates, this would lead to huge amounts of wasted bandwidth.
- separate ZIP files for each batch of new structures that were aggregated. However, this would mean much extra work for those who updated their version infrequently. Each ZIP file would have to be located and downloaded, which would likely lead to missed content on their part.
- the data as an RSS feed. This is reasonable, as it provides a method whereby one RSS file would summarise all the structures in CrystalEye at a given time, plus the timestamp at which those structures were aggregated. A suitable RSS reader could parse the feed and gather only those structures which were added since its last visit. However, even a summary of hundreds of thousands of structures would still give a feed of hundreds of megabytes in size, which would lead to a large amount of data having to be downloaded for even small updates.

### **Atom archived feeds**

The method chosen is a type of RSS feed, though one where it is not a monolithic document, but is split up into many smaller documents which:

...can be combined to accurately reconstruct the entries of a logical feed.[167]

This method is an *archived feed*, as defined in a proposed standard extension to the Atom protocol[166]. By splitting the feed up into smaller documents, an aggregator need only download small chunks of the Atom archive feed at a time before finding previously aggregated content. Thus the wasted bandwidth is minimised and sometimes removed completely. To implement this protocol, a Java library was written by the author.

For an archived feed, there is a fixed URL which points to the *current* document that contains the latest entries. In CrystalEye, this URL is

<http://wmm.ch.cam.ac.uk/crystaleye/feed/atom/feed.xml>

On 2008-03-10, this feed has the form:

```
<feed xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns="http://www.w3.org/2005/Atom">
  <title>CrystalEye: All Structures</title>
  <link rel="self"
        href="http://wmm.ch.cam.ac.uk/crystaleye/feed/atom/feed.xml"/>
  <subtitle>
    Feed summarising all the structures in CrystalEye.
  </subtitle>
  <updated>2008-03-08T05:27:09Z</updated>
  <author>
    <name>Chris Talbot</name>
  </author>
  <link rel="prev-archive"
        href="http://wmm.ch.cam.ac.uk/crystaleye/feed/atom/feed-4471.xml"/>
  <id>http://wmm.ch.cam.ac.uk/crystaleye/feed/atom/feed.xml</id>
  <entry>
    <title>
      Summary page for crystal structure from DataBlock
      d--mgdgroup-matthe~2-papers-zrc3-xrayre~1-test-k06mgd17
      in CIF b718678asup1 from article b718678a in issue 2008/11
      of Royal Society of Chemistry, Chemical Communications.
    </title>
    <link href="http://wmm.ch.cam.ac.uk/crystaleye/summary/rsc/cc/2008/11/data/
            b718678a/b718678asup1_d--mgdgroup-matthe~2-papers-zrc3-xrayre~1-
            test-k06mgd17/b718678asup1_d--mgdgroup-matthe~2-papers-zrc3-
            xrayre~1-test-k06mgd17.cif.summary.html"/>
    <id>urn:uuid:8cf20230-141d-41d2-830a-8724bc19ad8f</id>
    <summary>
      Summary page for crystal structure from DataBlock
      d--mgdgroup-matthe~2-papers-zrc3-xrayre~1-test-k06mgd17
      in CIF b718678asup1 from article b718678a in issue 2008/11
      of Royal Society of Chemistry, Chemical Communications.
    <updated>2008-03-08T05:27:09Z</updated>
    <link rel="enclosure"
          href="http://wmm.ch.cam.ac.uk/crystaleye/summary/rsc/cc/2008/11/data/
                b718678a/b718678asup1_d--mgdgroup-matthe~2-papers-zrc3-xrayre~1-
                test-k06mgd17/b718678asup1_d--mgdgroup-matthe~2-papers-zrc3-
                xrayre~1-test-k06mgd17.complete.cml.xml"
```

```

        type="chemical/x-cml"
        length="253229"/>
<link rel="enclosure"
      href="http://wmm.ch.cam.ac.uk/crystaleye/summary/rsc/cc/2008/11/data/
        b718678a/b718678asup1_d--mgdgroup-matthe~2-papers-zrc3-xrayre~1-
        test-k06mgd17/b718678asup1_d--mgdgroup-matthe~2-papers-zrc3-
        xrayre~1-test-k06mgd17_1.small.png"
        type="image/png"
        length="2389"/>
<content type="xhtml">
  <div xmlns="http://www.w3.org/1999/xhtml">
    
  </div>
</content>
</entry>
</feed>

```

Note that the entry content is linked via *enclosures*. The *current* feed always contains a *prev-archive* link which points to the immediately preceding archive document. This is usually the same name as the *current* document, but with a number indicating its position in the archive. In the above example it is:

<http://wmm.ch.cam.ac.uk/crystaleye/feed/atom/feed-4471.xml>

When creating archived feeds, a maximum number of entries per document is set. In CrystalEye this is 25, which means each document is around 30Kb large, the size of a reasonable webpage. Once the *current* feed has reached the limit, it is renamed to the same form as the *prev-archive* link, with the number incremented:

<http://wmm.ch.cam.ac.uk/crystaleye/feed/atom/feed-4472.xml>

and another archive document with no entries will be created at the *current* URL. Note that the documents other than the *current* document also contain two other links

- *next-archive* – points to the immediately following archive document,
- *current* – points to the *current* document.

The presence of these links allows aggregators to traverse the archived feed structure however they wish, and not just from newest to oldest.

An Open Source harvester for the CrystalEye Atom Archive feed has been written and blogged by Jim Downing[168]. The developers of the ChemSpider[89] site, a free access source of structure-based chemical information, have recently used this method to obtain the CrystalEye dataset. They were then able to extract all of InChIs from the data and use them to provide links to CrystalEye webpages from ChemSpider's own structure summaries.

### 4.3.7 CrystalEye data in RDF

Andrew Walkingshaw has, using his Golem[169] ontology system, created a service (currently unnamed) which takes the URL for a CrystalEye CML file, and converts it into RDF. For instance, providing it with the following URL:

```
http://wwmm.ch.cam.ac.uk/crystaleye/summary/acta/j/2002/02-00/data/
vi0154/vi0154sup1_tpp110kdata/vi0154sup1_tpp110kdata.complete.cml.xml
```

will return the RDF below (truncated for brevity):

```
<rdf:RDF
  xmlns:j.0="http://wwmm.ch.cam.ac.uk/crystaleye/dictionary#"
  xmlns:j.1="http://purl.org/dc/terms/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
>
  <rdf:Description rdf:about="http://wwmm.ch.cam.ac.uk/crystaleye/locations#
    00abbb71e196a8f7f2c0eeaec8c93c1e">
    <j.0:latitude rdf:datatype="http://www.w3.org/2001/XMLSchema#float">
      48.0833333
    </j.0:latitude>
    <j.0:longitude rdf:datatype="http://www.w3.org/2001/XMLSchema#float">
      -1.6833333
    </j.0:longitude>
    <j.0:address>
      "LCSIM UMR 6511 CNRS - Universit\\'e de Rennes 1,
      Institut de Chimie de Rennes, B\\^at. 10B,
      Campus de Beaulieu, Avenue du G\\'en\\'eral Leclerc,
      35042 Rennes, France "
    </j.0:address>
  </rdf:Description>
  ...
  <rdf:Description rdf:about="http://wwmm.ch.cam.ac.uk/crystaleye/summary/acta/
    j/2002/02-00/data/vi0154/vi0154sup1_tpp110kdata/
    vi0154sup1_tpp110kdata.complete.cml.xml">
    <j.0:_journal_name_full rdf:datatype="http://example.com/json">
      "Journal of Applied Crystallography"
    </j.0:_journal_name_full>
    <j.1:contributor>
      "Masson, Olivier"
    </j.1:contributor>
    <j.1:contributor>
```

```

    "Descamps, Marc"
  </j.1:contributor>
  ...
  <j.0:_publ_section_title rdf:datatype="http://example.com/json">
    "Ab initio structure determination of TriPhenyl Phosphite by powder
    synchrotron X-ray diffraction"
  </j.0:_publ_section_title>
  <j.0:oscarAnnotation
    rdf:resource="http://wwmm.ch.cam.ac.uk/crystaleye/ontology/ONT/Phosphite"/>
  <j.0:oscarAnnotation
    rdf:resource="http://wwmm.ch.cam.ac.uk/crystaleye/ontology/CM/Phosphite"/>
  ...
  <j.0:location rdf:resource="http://wwmm.ch.cam.ac.uk/crystaleye/locations#
    00abb71e196a8f7f2c0eeaec8c93c1e"/>
  ...
  <j.0:AcceptanceDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date">
    2002-01-09
  </j.0:AcceptanceDate>
</rdf:Description>
...
</rdf:RDF>

```

The generated RDF currently includes:

- bibliographic data,
- annotations for the OSCAR3 system[170],
- geographic data.

I have used an X/HTML metadata profile[171] to provide links to this RDF from the crystal structure summary pages in CrystalEye. For instance, the page at the URL below

```

http://wwmm.ch.cam.ac.uk/crystaleye/summary/acta/j/2002/02-00/data/
vi0154/vi0154sup1_tpp110kdata/vi0154sup1_tpp110kdata.cif.summary.html

```

contains the following code

```

<html>
  <head profile="http://purl.org/net/uriprofile/">
    ...
    <link rel="meta"
      href="http://www.cmlcomp.org/golem/demo/cedescribe/http%3A//
      wwmm.ch.cam.ac.uk/crystaleye/summary/acta/j/2002/
      02-00/data/vi0154/vi0154sup1_tpp110kdata/
      vi0154sup1_tpp110kdata.complete.cml.xml"
    />
  </head>
  <body>
    ...
  </body>
</html>

```

In the future, instead of linking to this RDF, it will be merged with the HTML as explicit RDFa[172].



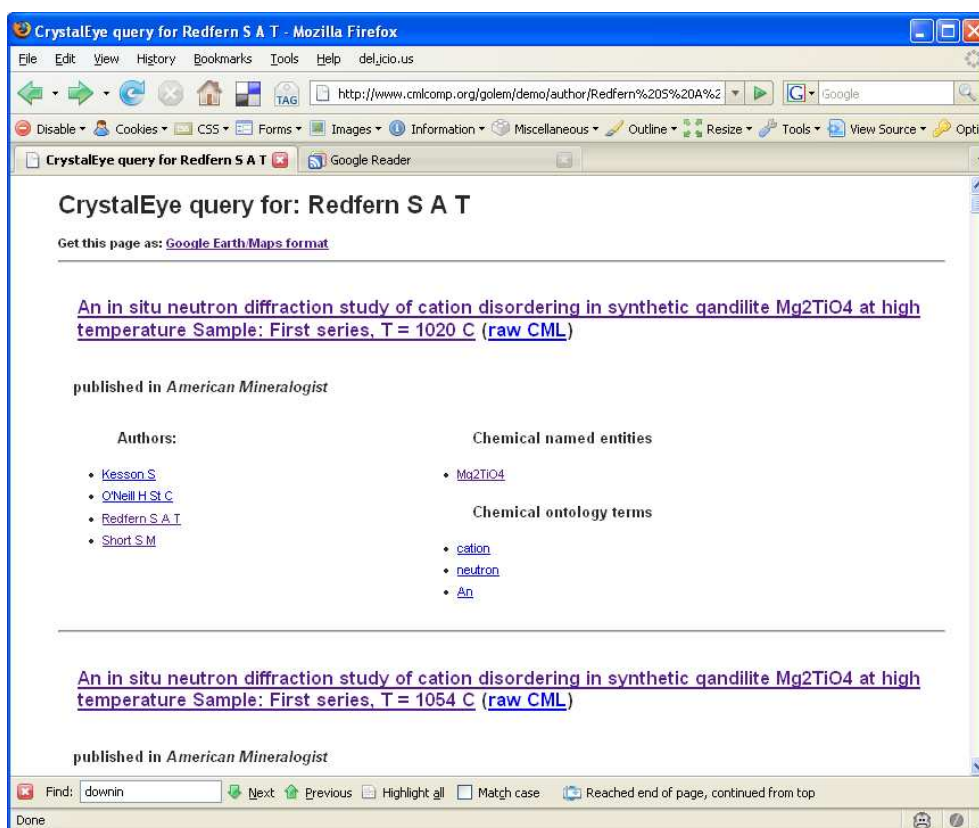


Figure 4.32: Results page for a SPARQL query of the CrystalEye RDF for all crystal structures authored by ‘Redfern S A T’.

Walkingshaw has created RDF for the entire CrystalEye dataset, and is experimenting with RDF-based search (see figure 4.32) and providing visualizations of the data (*e.g.* a video showing the geographic spread of Open crystallography[173]).

### 4.3.8 Further work

There is still much to do to improve the quality and rate of aggregation of data in CrystalEye. The following are viewed as important steps to this end.

## Self-deposition

At present CrystalEye aggregates almost entirely from the published literature. By providing a method for self-deposition into the system (either public or departmental), CrystalEye could start to collect some of the 80% of structures which are never published. Such a system has already been investigated in the SPECTRa project[130], and work has already begun to provide this functionality in CrystalEye.

## Community annotation and validation

It is inevitable that errors occur in any large dataset. Manual curation is extremely resource intensive and complete continuous curation of CrystalEye would require many hours work. Jim Downing and I have thus started to investigate *crowdsourcing*[174], so that interested users not directly involved in the project can aid curation of CrystalEye.

Community data checking is also being investigated at ChemSpider[175], and an experimental method based around Connotea[177] has been outlined[176]. Connotea is a social bookmarking tool aimed at scientists, which allows URLs to be tagged and shared by members. Our mechanism suggests that on finding a bug in a CrystalEye document, the user should bookmark that URL in Connotea with the tag `crystaleyeproblem`, where the Description field can be used to describe the problem. By subscribing to the RSS feed at, <http://www.connotea.org/tag/crystaleyeproblem>, we can then be instantly notified of any bugs found in CrystalEye. Once the bug is fixed, the same URL can be bookmarked with another tag, `crystalfixed`.

Another potential use of crowdsourcing in CrystalEye is for annotation of data to aid processing. It is accepted that programs cannot be created to perform all the tasks an expert in a particular field is capable of, such as:

- understanding data that deviates slightly from its specification,
- processing data in areas where the rules and semantics are not well-defined (*e.g.* finding a set of BOACs for an organometallic skeleton).

There are many documents that CrystalEye is unable to completely process by virtue of the data being either invalid or ambiguous. By providing these documents to human experts in a suitable Web UI, missing or erroneous data could be corrected, and a higher proportion of the data aggregated could be processed to completion, and thus be made available to the community.

## 4.4 Conclusions

CrystalEye provides a completely automated method for gathering, processing and disseminating semantic Open crystallographic data found on the Web. The software comprising CrystalEye has been well tested on crystallographic data from over 85,000 CIFs. This has made CrystalEye a robust service, which has been running for the first six months of 2008 without requiring human intervention.

All data is made Openly available via a website, which utilises numerous Web 2.0 technologies for dissemination, as well as enabling chemical search by exposing the InChI and SMILES for each structure. Methods for expanding and improving CrystalEye have been discussed, with work already having started in some areas. CrystalEye is also involved in funded projects due to start in the latter half of 2008.

## Chapter 5

# High-throughput prediction of solid-state geometry using semi-empirical methods

### 5.1 Introduction

The availability of CrystalEye provided access to a large, high-quality set of crystallographic data which could be used to perform e-Science. Townsend previously used molecular structures from CrystalEye to assess the ability of different computational methods to predict the geometry of structures in the gas-phase[165]. One of the packages Townsend used in his work was MOPAC, a general-purpose semi-empirical molecular orbital package. He found, for molecules containing first and second row elements, that the PM3 method in MOPAC gave a good approximation of the structure in the majority of cases. However, certain structural fragments (*e.g.* N-N bonds) gave systematic errors. He was then able to report back to Stewart, the author of MOPAC, those molecular sub-systems that appeared to need further parameterization.

In March 2007 I met with Stewart to discuss how I might be able to extend Townsend's work. Stewart informed me of the latest additions to MOPAC (the current version is MOPAC2007), which included (then unpublished) work on:

1. a new parameterization, PM6 (now published[178]).

2. new functionality that allows the calculation of solid-state geometry (now published[179]).

The aim of Stewart’s work on PM6 was to carry out a systematic global parameter optimization of all the main group elements, with emphasis on compounds of interest in biochemistry; and to extend the methodology by performing a restricted optimization of parameters for the transition metals. As a result, PM6 contains atom and atom-pair parameters for 70 elements (all main group elements up to Bismuth).

Despite the addition of functionality to calculate solid-state geometry, no solids were used in the training or survey sets whilst parameterizing PM6. Stewart stated that this was because of the lack of computer power available, and had since performed around 100 geometry optimizations on both organic and inorganic crystal structures. As I had access to CrystalEye and Cambridge University’s computing grid, CamGrid, I had the possibility of applying MOPAC’s solid-state capability in a high-throughput manner to a large number of structures. By selecting structures containing a wide range of systems, this would provide a good test of the method, as well as its use for data validation.

Townsend’s work on geometry prediction in the gas-phase highlighted various molecular fragments and atom-pairs that were not modeled well in the PM3 parameterization. During the work, I can expect to find similar results, and, as no solids were used during parameterization for PM6, I can also expect to find poorly parameterized intermolecular interactions leading to a variety of distortions of the crystallographic cell (discussed further at the start of section 5.5).

To undertake this work, as the amount of data involved is large (several gigabytes in many thousand files), it is preferable to automate as much as possible of the workflow. Not only this, but in order to minimise format conversion errors and keep data extraction simple, CML must be used wherever

possible. For this reason, the incorporation of CML output functionality into MOPAC was first investigated.

## 5.2 MOPAC-CML

MOPAC, like many computational chemistry programs is written in Fortran, an imperative programming language that is especially suited to numeric computation. Unlike the numerical support in Fortran, the I/O facilities are very limited, where reading and outputting much more than plain text is difficult. As such, each program has developed its own idiosyncratic input and output formats, which generally consists of whitespace delimited ASCII text designed primarily for human readability (see figures 5.1 and 5.2).

It is far more difficult to write bug-free programs to parse these files (as Townsend described in his efforts to post-process program output using XML stylesheets[165]) when compared to highly structured formats such as XML. Thus, automating the analysis and reuse of the data produced by these programs is difficult and error-prone. Ideally, computational chemistry programs should read and output structured, semantically-rich XML, though until recently the lack of an XML library for Fortran meant this was not feasible.

### 5.2.1 FoX

As part of the eMinerals project[180], investigations have been undertaken into the use of CML for data exchange, visualization and storage for computational science[181]. To facilitate these aims, the open source **FoX** (Fortran XML) library has been developed by White[182].

**FoX** is the first fully-validating XML parser and writer to be written in Fortran 95, and can be used directly within programs rather than having to post-process existing output. XML output is obtained by successive calls to functions named `xmlNewElement`, `xmlAddAttribute` and `xmlEndElement` where state is kept by the library to ensure well-formedness throughout the document. There are also a series of additional function calls to directly

```

MERS=(2,1,2) GNORM=1 T=172800 DUMP=172801 LET DDMIN=0.0
Albite

Si    0.00000000 +1    0.00000000 +1    0.00000000 +1    0.0000
Na   -2.85299900 +1   -4.37400000 +1    2.20700000 +1    0.0000
Na    1.19399900 +1    2.06100000 +1    2.20600000 +1    0.0000
Na    1.68000200 +1   -4.44100000 +1    0.52000000 +1    0.0000
Na   -2.42599900 +1    1.99300000 +1    0.52000000 +1    0.0000
Al   -2.11299900 +1   -1.93000000 +1   -3.62000000 +1    0.0000
O    -2.25899900 +1   -0.34700000 +1   -3.22500000 +1    0.0000
O    -1.13399900 +1   -2.68600000 +1   -2.52300000 +1    0.0000
O   -3.59099900 +1   -2.65200000 +1   -3.72000000 +1    0.0000
Si   -1.13099900 +1    0.74300000 +1   -2.73900000 +1    0.0000
Si   -0.98199900 +1   -2.67300000 +1   -0.88000000 +1    0.0000
O    -1.41099900 +1   -2.21100000 +1   -5.08100000 +1    0.0000
O    -0.45299900 +1   -4.16300000 +1   -0.40300000 +1    0.0000
O   -2.41399900 +1   -2.32800000 +1   -0.19400000 +1    0.0000
O    0.14600100 +1   -1.58300000 +1   -0.39400000 +1    0.0000
Si   -3.98799900 +1   -1.91700000 +1   -0.09000000 +1    0.0000
Al   -1.11799900 +1   -5.65200000 +1   -0.66900000 +1    0.0000
Si    1.93400100 +1    4.50400000 +1   -3.62000000 +1    0.0000
O    1.78900100 +1    6.08700000 +1   -3.22500000 +1    0.0000
O    2.91400000 +1    3.74800000 +1   -2.52300000 +1    0.0000
O    0.45600100 +1    3.78300000 +1   -3.72000000 +1    0.0000
Si   -0.19199900 +1   -6.14200000 +1    3.60700000 +1    0.0000

.....

```

Figure 5.1: An example MOPAC input file to calculate the solid-state geometry of Albite. The input parameters (top line), and a portion of the Z-matrix (below) are shown.

```

GEOMETRY OPTIMISED USING EIGENVECTOR FOLLOWING (EF).
SCF FIELD WAS ACHIEVED

PM6      CALCULATION      MOPAC2007 (Version: 7.148W)
                        Tue Jul 24 07:59:50 2007

FINAL HEAT OF FORMATION = -15415.68085 KCAL = -64499.20869 KJ
H.o.F. per unit cell = -963.48005 KCAL, for 16 unit cells

TOTAL ENERGY = -42176.06447 EV
ELECTRONIC ENERGY = -202090430.01383 EV
CORE-CORE REPULSION = 202048253.94936 EV

IONIZATION POTENTIAL = 5.98474
NO. OF FILLED LEVELS = 512
MOLECULAR WEIGHT = 4195.568

MOLECULAR DIMENSIONS (Angstroms)
Atom      Atom      Distance
O 189 O 27 25.45197
O 97 Si 126 18.42300
O 182 Si 36 15.66552

VOLUME OF UNIT CELL 2836.350 CUBIC ANGSTROMS
DENSITY 2.456 GRAMS/CC
152 a. b. c. alpha. beta. gamma: 8.772 12.694 7.038 89.71 64.80 90.35 Vol: 709.09 Density: 2.456 HoF: -963.480 Grad: 0.25
Pressure required to constrain translation vectors
Tv( 209) Pressure: 0.00 GPa
Tv( 210) Pressure: -0.01 GPa
Tv( 211) Pressure: 0.00 GPa

SCF CALCULATIONS = 153
COMPUTATION TIME = 15 HOURS 6 MINUTES AND 20.906 SECONDS

```

Figure 5.2: A portion of the MOPAC output file from the calculation of the solid-state geometry of Albite.

translate native Fortran data structures to appropriate collections of XML structures.

In addition, FoX also provides the **WCML** library, which allows developers to output CMLComp[183], a subset of CML. Methods are provided to output commonly-grouped collections of CML elements, for instance:

```
call cmlAddMolecule(xf=xmlFile, coords=array_of_coords,
elems=array_of_elements)
```

would output the following chunk of CML:

```
<molecule>
  <atomArray>
    <atom x3='0.1' y3='0.1' z3='0.1' elementType='H' />
    ....
  </atomArray>
</molecule>
```

Similar interfaces are provided for all portions of CML commonly used by simulation codes.

Many computational atomic molecular and molecular physics codes are Open Source or are available for community modifications, and most of those used in the eMinerals project have already been modified with the **WCML** interface. These include the public releases of SIESTA 2.0 and DLPOLY 3, as well as the version of CASTEP[184] being developed by Materials Grid[185].

### 5.2.2 MOPAC and FoX

Following discussions with Stewart, it became clear that there were two options to add the FoX functionality into MOPAC to output CML:

1. Include FoX method calls throughout the code to mirror those used for the standard text output.
2. Have all standard output also directed to a new module, which can then be modified to use FoX to output CML.



The former option would mean that, as **FoX** calls would be scattered throughout the whole of the code, **FoX** would always be required when building the system. Stewart preferred not to do this, or to maintain two different versions of MOPAC for those who do or do not wish to output CML, and so suggested that the latter method was preferable. With this option, to create a MOPAC executable that outputs CML, users can replace one Fortran module and use a slightly altered make file to include the **FoX** libraries.

As a result, Stewart added a new module called **to\_screen** and altered all those modules that contain calls to standard output so that there is a duplicate call to **to\_screen**. For instance, in the **writmo** module, the call to output the final heat of formation to the standard output is:

```
write (iw, &
' (4/10X, 'FINAL HEAT OF FORMATION = ', F17.5, ' KCAL' , ' = ', F14.5, ' &
&KJ' )) escf, escf*4.184D0
```

There is now a call to **to\_screen** shortly after:

```
write (line, '(10x,a,f16.5,a)') "Final heat of formation = ", escf, " kcal/mol"
call to_screen(line)
```

When **to\_screen** is run, it uses the message passed as a parameter to decide what needs to be output.

The **to\_screen** module contains all the necessary control structures and example outputs to reproduce the standard textual output of MOPAC. Thus I could use it as a template and simply modify the provided calls to use calls to **FoX**. For instance, the call for outputting the heat of formation was originally:

```
write(hook, "(a,sp, d13.6,a)") " HEAT_OF_FORMATION:KCAL/MOL=", escf
```

but I have now changed it to:

```
! write(hook, "(a,sp, d13.6,a)") " HEAT_OF_FORMATION:KCAL/MOL=", escf
call cmlStartElement(xf=cmlfile, name='scalar')
  call xml_addAttribute(xf=cmlfile, name='dictRef', value='mopacDict:escf')
  call xml_addAttribute(xf=cmlfile, name='dataType', value='fpx:real')
  call xml_addAttribute(xf=cmlfile, name='units', value='units:kcal.mol-1')
  call xml_addCharacters(xf=cmlfile, chars=escf)
call xml_EndElement(xf=cmlfile, name='scalar')
```

so that the output is now a CML **scalar** element with the appropriate dictionary reference, data type and units, such as:

```
<scalar dictRef="mopacDict:escf" dataType="fpx:real"
      units="units:kcal.mol-1">-1.541568085299e4</scalar>
```

By doing this for the entire module, all of the data produced by MOPAC can be captured as appropriate CML elements to create a valid document. Note that the original output calls have all been maintained, but have been commented out.

An executable of the modified version of MOPAC was compiled, along with FoX v2.0.2, using G95 v0.9 on a SUSE Linux machine with an i686 CPU architecture. Using this executable, when a calculation with an input file `<id>.mop` is run, the standard output, as usual, is written to `<id>.out`, but there is also a file written to `<id>.xml` containing the output CML.

### 5.2.3 MOPAC-CML and Jmol

Another benefit of having CML output direct from MOPAC is that it enables the viewing of renderings of the data immediately in current chemical visualization tools, rather than first having to convert to a viewer-friendly format. I have altered the `to_screen` module so that the geometry of the structure is output as a **molecule** element every 25 cycles (see figure 5.3), thus allowing the tracking of atomic movement during optimization. When these files are viewed in Jmol, the starting geometry will be displayed initially, and by using the scroll buttons, a user can click through each intermediate structure until the final geometry is displayed. This combination is very useful for viewing and analyzing atomic movement during rearrangements (as in figure 5.4).

## 5.3 High-throughput computing

High-throughput computing (HTC) is the use of many computing resources over long periods of time to accomplish a computational task\*, often involv-

---

\*As opposed to high-performance computing (HPC), where the emphasis is placed on a much shorter timespan (FLOPS (FLoating-point Operations Per Second) is the common

```

.....truncated.....
<module title="cycle_49">
  <scalar dictRef="mopacDict:escf" dataType="fpx:real" units="units:kcal.mol-1">-1.245266358454e4</scalar>
  <scalar dictRef="mopacDict:gnorm" dataType="fpx:real" units="units:kcal.mol.[Åring]">
  4.123735205456e1</scalar>
  <property dictRef="mopacDict:tvec" title="TRANS_VECTS_UPDATED" id="tvec">
    <matrix rows="3" columns="3" dataType="fpx:real" units="units:[Åring]">1.420796087853e1
    -7.719269450956e-3 3.244430058757e-2 2.024955619872e-1 1.724810044121e1 3.333254413991e-2
    -6.680515605748e-2 -7.298776017493e-2 1.319505449135e1</matrix>
  </property>
</module>
<module title="cycle_50">
  <scalar dictRef="mopacDict:escf" dataType="fpx:real" units="units:kcal.mol-1">-1.245086607676e4</scalar>
  <scalar dictRef="mopacDict:gnorm" dataType="fpx:real" units="units:kcal.mol.[Åring]">
  1.304758982337e2</scalar>
  <molecule id="geom-step">
    <atomArray>
      <atom elementType="Mg" x3="-4.746606333571" y3="-6.739955338035" z3="7.26121682754" id="a1"/>
      <atom elementType="S" x3="2.264915167977" y3="1.514574739963" z3="1.721988683499" id="a2"/>
    </atomArray>
    <truncated.....>
  </molecule>
  <property dictRef="mopacDict:tvec" title="TRANS_VECTS_UPDATED" id="tvec">
    <matrix rows="3" columns="3" dataType="fpx:real" units="units:[Åring]">1.420493820110e1
    -1.181906919486e-2 3.478056408520e-2 2.033119537466e-1 1.725485390831e1 3.612830244442e-2
    -6.620961700845e-2 -7.697429962263e-2 1.318371829058e1</matrix>
  </property>
</module>
<module title="cycle_51">
  <scalar dictRef="mopacDict:escf" dataType="fpx:real" units="units:kcal.mol-1">-1.245751953996e4</scalar>
  <scalar dictRef="mopacDict:gnorm" dataType="fpx:real" units="units:kcal.mol.[Åring]">
  4.601185309391e1</scalar>
  <property dictRef="mopacDict:tvec" title="TRANS_VECTS_UPDATED" id="tvec">
    <matrix rows="3" columns="3" dataType="fpx:real" units="units:[Åring]">1.420283083551e1
    -1.505315776393e-2 3.669328313197e-2 2.040705895703e-1 1.725985417844e1 3.781985113598e-2
    -6.484221349925e-2 -7.662317540970e-2 1.317737896728e1</matrix>
  </property>
</module>
.....truncated.....

```

Figure 5.3: Section of a MOPAC-CML output file showing the details of successive optimization cycles. Every 25<sup>th</sup> cycle the geometry is output, as shown in the module for cycle 50.

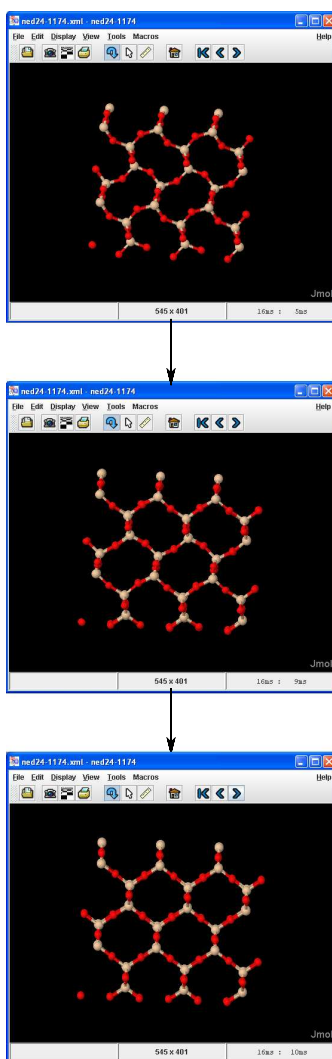


Figure 5.4: Images showing the use of Jmol to view the MOPAC-CML output for a geometry optimization of low-cristobalite. The change in geometry can be viewed using the scroll buttons in the top-right of the application window. Calculations on the phases of silica are discussed further in section 5.6.2.

ing grid computing technologies and techniques. HTC is particularly useful where the same analysis needs to be applied to independent sets of data repeatedly; as is the case with using the same application to optimise the geometry of a large set of crystal structures.

### 5.3.1 Condor

Condor[186] is an Open Source, high-throughput computing software framework. Condor may be used to network machines into clusters of compute nodes (called pools), and allows users to submit jobs to these resources. Executable files may be provided along with input files so that jobs involving any application can be run (providing the application can run on the architectures in the cluster). Condor pools may consist of dedicated nodes, as well as workstations, which are used at times when they would otherwise be idle. Condor's *flocking* technology[187] can be used to combine pools to build Grid-style computing environments that cross administrative boundaries (as in CamGrid below).

Condor provides features commonly found in batch scheduling systems, such as job queueing and scheduling policies, priorities, resource monitoring and resource management. Condor places submitted jobs into a queue, decides where and when to run them, monitors jobs as they run and returns the results back to the user.

### 5.3.2 CamGrid

CamGrid[37] is a project at the Cambridge e-Science Centre[36] to provide a computational grid for the university's members. It was initially conceived by a number of departments and groups within the university who decided to federate their computational facilities. CamGrid is based on Condor, with each institution maintaining its own Condor pool (there are currently 12). The resources of all participating pools are shared using Condor's flocking mechanism. The Unilever Centre for Molecular Science Informatics (UCC)

---

yardstick in HPC).

provides a pool consisting of 44 dedicated Condor hosts and a central manager from which jobs can be submitted[188].

## 5.4 The calculation workflow

The steps necessary to perform the calculations and obtain the output are:

1. select suitable structures from CrystalEye,
2. create MOPAC input files from each CML file,
3. create a Condor submit file for each MOPAC input file,
4. upload the MOPAC input and Condor submit files to the Condor submit host,
5. submit the jobs to the Condor pool,
6. retrieve the output CML files.

I aimed to automate as much of this process as possible. The discussion and implementation of each step is provided below.

### 5.4.1 Structure selection

The fact that I have access to an Open, CML-based dataset of structures in CrystalEye makes data aggregation and filtering simpler than the methods previously available, as:

- I don't have to pay or ask permission to get or reuse the data,
- a harvesting method and tool is provided (section 4.3.6),
- I can use existing tools to parse, extract and perform complex filtering of the CML data provided.

After harvesting the CrystalEye dataset onto a local machine, I then created a Java program that would iterate through all of the CML files, and use XPath queries to find suitable structures. The precise details of the filters used for both organic and inorganic structures are described in section 5.5

### 5.4.2 MOPAC input

Unlike more conventional methods, MOPAC does not normally use a fundamental unit cell. Neither does it sample the Brillouin Zone in order to model the electronic structure. Instead, it uses a large unit cell, called a *cluster*, and applies the Born-von Karman[189, 190] periodic boundary conditions. The online MOPAC manual states:

The unit cell must be large enough that an atomic orbital in the center of the unit cell has an insignificant overlap with the atomic orbitals at the ends of the unit cell. In practice, a translation vector of more that about 7 or 8Å is sufficient.[191]

Stewart provided me with a set of example input files for the solid-state calculations. These examples contained a common set of input parameters which I adopted, though I also added a time limit and a flag indicating that no checkpoint file need be written during calculation (to save space on the submit host). An example input file is shown in figure 5.5, the first line of which contains the set of parameters that were used throughout this work. The definitions for these are:

**MERS=(1,1,2)** Used to signify the number of unit cells used along the a, b and c axes respectively. Using the recommendation from the MOPAC manual above, I used a minimum translation vector of 7Å for the clusters in the calculations. As many structures have unit cell parameters smaller than this, the calculation must be performed on a supercell of the structure.

**GNORM=5** Signals that geometry optimization should exit as soon as the gradient norm drops below 5 kcal mol<sup>-1</sup>Å<sup>-1</sup>, corresponding to an uncertainty in the optimized geometry of about 0.001Å.

```

MERS=(1,1,2) GNORM=5 T=172800 DUMP=172801 LET DDMIN=0.0
Apatite

Ca      0.09657818 +1    -0.1493619 +1    -0.0334363 +1    0.0000
Ca     -5.16249629 +1     1.3106236 +1     0.0527765 +1    0.0000
Ca     -0.12184572 +1    -0.0605356 +1     3.4045161 +1    0.0000
Ca     -4.92446544 +1     1.2579660 +1     3.5455080 +1    0.0000
Ca     -3.22836347 +1    -1.6284472 +1    -1.6823064 +1    0.0000
Ca     -1.47446444 +1    -3.2009326 +1     1.6876740 +1    0.0000
Ca      0.25802281 +1     3.5166313 +1    -1.8976863 +1    0.0000
Ca      3.44817233 +1     0.3782045 +1     1.7582007 +1    0.0000
Ca      3.06074056 +1    -1.9916192 +1    -1.6619276 +1    0.0000
Ca     -1.86370185 +1     2.9242263 +1     1.7421366 +1    0.0000
F       5.30564757 +1    -1.2555191 +1    -1.6900106 +1    0.0000
F       5.08337702 +1    -1.0886140 +1     1.4948603 +1    0.0000
O      -1.58676950 +1    -0.7832258 +1     1.6834229 +1    0.0000
O       0.14383059 +1     1.5583528 +1     1.5026995 +1    0.0000
O       1.50090699 +1    -1.0555605 +1     1.8520239 +1    0.0000
O      -3.59038848 +1     2.0005361 +1    -1.7266213 +1    0.0000
O       3.49737368 +1     2.2005859 +1    -1.6992300 +1    0.0000
O       0.08144978 +1    -4.0903926 +1    -1.7604481 +1    0.0000
O      -3.40004295 +1     1.1966591 +1     1.6606886 +1    0.0000
O       2.50203081 +1     2.4567874 +1     2.1232987 +1    0.0000
..... Z-matrix truncated .....
Tv      8.81832335 +1     2.6251886 +1    -0.1846344 +1
Tv      6.67869939 +1    -6.3683740 +1     0.0024453 +1
Tv     -0.12178830 +1     0.0094759 +1    -13.8500754 +1

```

Figure 5.5: Example input file for a MOPAC solid-state calculation for Apatite. Note that the Z-matrix has been truncated for brevity.

LET DDMIN=0.0 During a calculation, the confidence level or trust radius is continuously checked. If this becomes too small, the calculation will be stopped. This can readily happen if the geometry was already almost optimized (as may be the case with the starting experimental geometries). In these cases, this combination of two keywords is to be used[192].

T=172800 Sets a CPU time limit of 172800 seconds (two days) for the calculation.

DUMP=172801 If this keyword is not provided, restart files are written automatically at two hour CPU time intervals to allow long jobs to be restarted if they are terminated catastrophically. The time for this keyword has been provided so that the interval for writing the restart file is longer than the maximum calculation time.

The next two lines are used for calculation comments. Below this is the Z-matrix for the cluster, where the element symbol is followed by the Cartesian



coordinates for the atom. Stewart stated that in his work:

“An attempt was made initially to use internal coordinates, but the numerical instabilities associated with the geometric gradients at the interfaces of the unit cells rendered their use impractical.”[179]

thus I used Cartesian coordinates exclusively. At the end of the Z-matrix are the translation vectors for the cluster, denoted by the symbol Tv. The +1 next to each coordinate signals that it should be optimized. During Stewart’s work all unit cell parameters were optimized, as were the coordinates of all atoms within the unit cell. Thus, during the calculations, I will do the same.

The `legacy2cml` library[114] contains the `MOPACINConverter` class that allows the conversion of CML molecules to MOPAC input files for molecular calculations. This converter was extended to allow users to use molecules containing `crystal` elements to create MOPAC solid-state input files. An example of the use of this converter is shown below:

```
String[] args = {"-INFILE", "c:/path/to/cml/file.cml",
                "-OUTFILE", "c:/output/file/path.mop",
                "-MAKELEGACY",
                "-KEYWORDS", "GNORM=5",
                "-TITLE", "ID",
                "-JOBID", "1",
                "-MINTV", 7,
                "-CRYSTAL"};

};
MOPACINConverter mopConverter = new MOPACINConverter();
mopConverter.runCommands(args);
```

The options used in the above example are:

- `INFILE` – the path to the input CML file.
- `OUTFILE` – the path to the output MOPAC input file.
- `MAKELEGACY` – indicates to the parser that it should create MOPAC input files from CML, and not the other way round (which the class is able to perform).
- `KEYWORDS` – the job parameters (first line for the input file).

- **TITLE** – the job title (second line). In this work this is used to indicate the origin of the crystal structure, including the publisher, journal, year, issue, article and CIF datablock information, *e.g.*

`acta_b_1992_06-00_as0598sup1_as0598b`

- **JOBID** – the unique ID of the job (third line). This is useful in case more than one job needs to be run on any structure.
- **MINTV** – the minimum translation vector length of the generated supercell.
- **CRYSTAL** – indicates that the input file is for a solid-state calculation.

When **MOPACINConverter** is executed, the following steps are performed:

1. the CML file is read into an in-memory **XOM** representation,
2. the cell parameters from the **crystal** child of the **molecule** element are extracted,
3. the size of the supercell needed to fulfill the minimum cluster size requirements is calculated,
4. the cluster is generated from the **molecule** (methods were added to the **CrystalTool** class of the **JUMBO** library to allow this),
5. for each atom, the Cartesian coordinates are calculated from the fractional coordinates provided,

Using the input parameters and IDs provided, as well as the generated coordinates, the input file can then be created and output.

### 5.4.3 Condor input

An example file for the submission of a MOPAC job to Condor is shown in figure 5.6. Assuming the file is named `ned24-2392.mop`, and that the user has a shell open and pointing at the same directory that file is stored in, the following command would submit the job from the host to the pool:

```

universe = Vanilla
requirements = Arch == "X86_64" && OpSys == "LINUX" && Memory > 500
executable = ../MOPAC2007.exe
log = ned24-2392.log
log_xml = true
periodic_remove = TARGET.ImageSize >2000000
error = ned24-2392.err
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
transfer_input_files = ned24-2392.mop
Arguments = ned24-2392.mop
Queue 1

```

Figure 5.6: Example submit file for the submission of a MOPAC job to Condor.

```
condor_submit ./jobs/ned24-2392.condor.sh
```

Here follows a description of each line in the submit file:

1. the Condor execution environment
2. the operating system, CPU architecture and minimum memory required for a compute node to be able to run the job
3. the path on the submit host to the executable file to be used for the job
4. the path that the job log should be written out to on the submit host
5. flag that the log file should be in XML
6. flag that the job should be terminated if it takes up more than 2Gb drive space
7. the path that the output of *stderr* should be written to on the submit host
8. flag that all files necessary to start the job should be transferred from the submit host to the Condor pool (the executable is always transferred when this flag is used)

9. flag that files should only be written from the pool to the submit host once the job has terminated
10. path of the input file to be transferred to the pool
11. the argument needed in the job execution command
12. tells Condor to queue the job for execution

#### 5.4.4 Job submission and retrieval

All job submissions to CamGrid must be done via a Condor submit host (during this work, it is the UCC host, `hadean--ch.grid.private.cam.ac.uk`). Unfortunately there are no methods available for automating the submission of jobs from workstations, so all the MOPAC input and Condor submit files must be copied to `hadean` and run from there. This was achieved by a shell script.

Another shell script was then written to submit the jobs from the host in batches of 500. I kept track of job termination, and when each batch had finished, the next was submitted. Once all the jobs had terminated, all output was archived on `hadean`, with copies of the CML output being downloaded onto a workstation for analysis (again achieved by a shell script).

#### Condor-related job failures

Prior to the execution of the batch jobs, a number of tests were run to ensure that running MOPAC jobs on Condor would be successful. These tests uncovered two Condor-related problems which caused jobs to fail.

Initially, each Condor submit files contained commands to indicate the names of the output files on each calculation (as below):

```
input = ned24-1396.mop
output = ned24-1396.out
```

This is necessary for many applications, though in MOPAC only the input file needs to be specified. The name of the output file is that of the input file, but with the `.out` MIME (and also `.xml` in the modified version) replacing `.mop`. As I was instructing Condor to create files that MOPAC was also writing to, all submitted jobs failed immediately.

Many of the machines in each Condor pool contain more than one VM (virtual machine), where each is able to run a separate job. It was noted that when a machine was running more than one job from a particular user, if one job terminated (whether successfully or not), then the other jobs would be terminated at exactly the same time. This was discussed with Mark Calleja (manager of CamGrid), who identified that when a job terminated, a SIGTERM was automatically being sent to all processes owned by that user. This was meant to clean up any unclosed processes from the terminated job, but as a result was also closing any other jobs owned by the user. This problem has subsequently been fixed.

#### **5.4.5 The overall workflow**

The workflow for the high-throughput solid-state calculation of MOPAC jobs is shown in figure 5.7. For each step, programs and scripts were written to automatically process arbitrarily large datasets, though they were executed manually rather than the ideal of being linked together in an automated workflow. It would be technically possible to automate the whole process, though the lack of an interface for automated job submission to CamGrid would require a workaround.

### **5.5 The first protocol**

By using structures from CrystalEye as starting points for the calculations, I can investigate how closely MOPAC calculations agree with experimentally observed structures. Ideally all structures would remain unchanged during calculation, though in reality I may expect the following changes to occur:

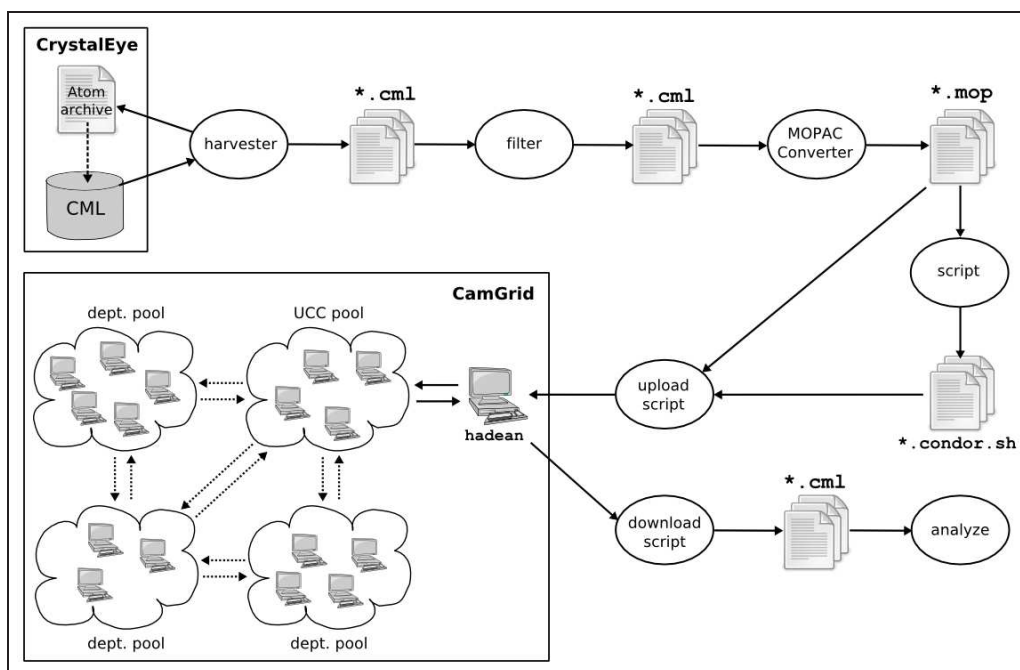


Figure 5.7: The workflow for performing high-throughput solid-state MOPAC calculations. CML is harvested from CrystalEye via the Atom archive feed, before being filtered by a Java program based around XPath queries. The remaining CML files are then parsed by the `MOPACINConverter` Java class into solid-state MOPAC input files, which are subsequently converted into Condor submit files. Both of these are then uploaded to the Condor submit host, `hadean`, via a shell script. Batches of jobs are submitted from `hadean` to the UCC Condor pool, which may communicate with other departmental pools to execute the jobs. Once terminated, Condor will return the output files to `hadean`, which are downloaded to a workstation for analysis via another shell script.

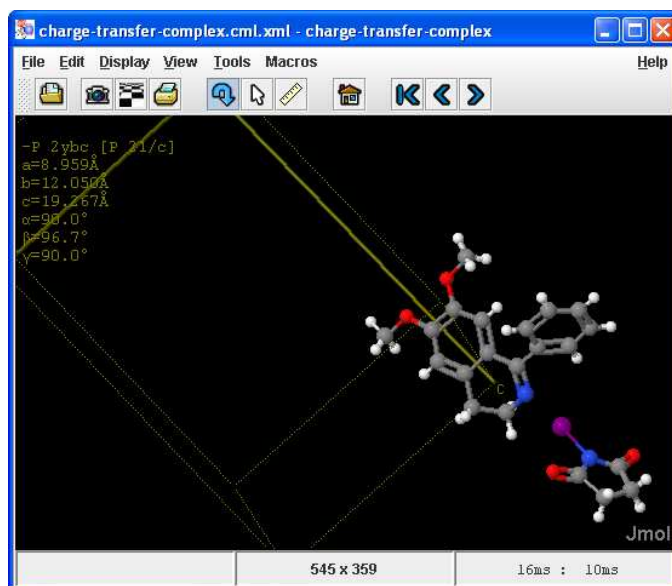


Figure 5.8: Figure showing an experimentally observed charge-transfer complex of *N*-Iodosuccinimide and an imine[194]. The intermolecular N-I distance is 2.5Å, and the intramolecular distance is 2.1Å.

- expansion/contraction — many of the structures obtained from CrystalEye will have been observed at various temperatures or pressures. As MOPAC calculates all structures at the same temperature (0K), I can expect some expansion or contraction to occur.
- rotation of subspecies — in some structures, large rotation of subspecies and polyhedra can occur at relatively low energies.
- phase changes — MOPAC may rearrange the structure into one that it evaluates to be a more energetically favourable form.
- formation of new bonds — particularly between electron-rich and electron-poor atoms. For instance, donor-acceptor (charge-transfer) complexes involving molecular halogens, interhalides and a wide range of covalent halides are well known[193] (as in figure 5.8). In these complexes, intermolecular distances between the two atoms taking part in the charge-transfer are considerably smaller than the sum of the van der Waals

radii, and in some cases, approach the sum of the covalent radii.

- mobile protons — Townsend discussed intramolecular H movement between electronegative atoms in his work on PM3. I can also expect intermolecular H movement in the solid-state, particularly where H-bonding occurs.

It is very difficult to create a method to identify interesting changes in crystal structures; Stewart proposed using density as the most obvious scalar property to observe structural change.

### 5.5.1 Organic structures

The selection of organic structures from CrystalEye was performed in sets of 500, which would subsequently be filtered before the file conversions and job execution took place. Once one set of calculations had finished, the process would then be repeated on the next set of 500 structures. The total number of organic structures selected by the end of the process was 6500. This included all those organic structures from Acta Cryst. Section A and Section B, as well as all those from Section C up to the end of 1998.

From the 6500 selected structures, all of those containing disorder, as indicated by the author of the CIFs, were removed, leaving 5916 structures. From these, all those without an author provided formula (usually obtained from the `_chemical_formula_sum` element), which is required for the validation of the the composition of the structure, were removed; this left 5798 structures.

For the remaining 5798 structures, the calculation workflow was performed as described previously in section 5.4. Of these calculations:

- 4245 converged successfully,
- 875 reached the job time limit before converging,
- 651 reached the maximum cycle limit before converging,



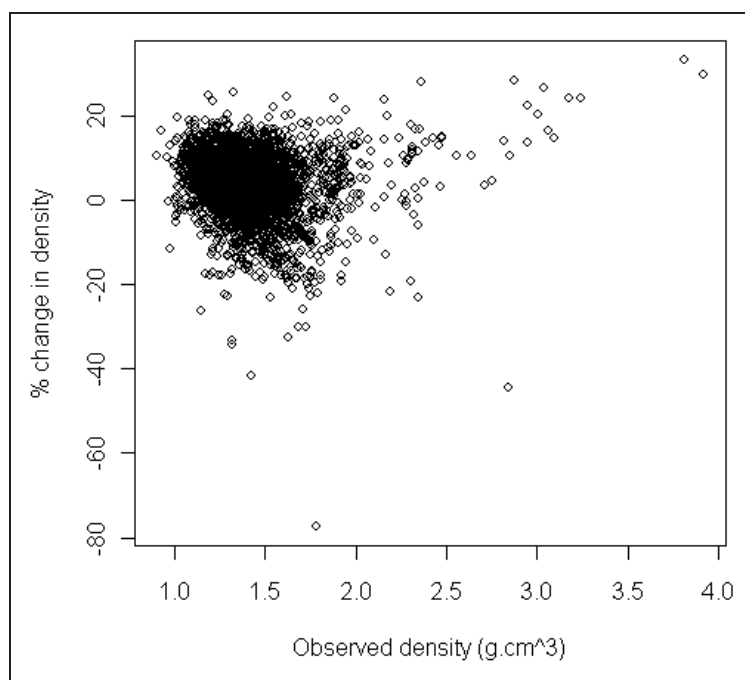


Figure 5.9: Plot showing the experimentally observed densities against the change in density predicted by MOPAC for the set of 4243 converged organic structure calculations.

- 27 resulted in MOPAC errors that gave controlled abortions,
- 0 resulted in an uncontrolled job abortion.

For those calculations that converged successfully, figure 5.9 shows a plot of the experimentally observed crystal density against the change in density predicted by MOPAC. The average unsigned and signed errors in calculated density for this dataset are 6.4% and 4.0% respectively. Before any further analysis of the results were undertaken, those calculations for inorganic structures were run.

### 5.5.2 Inorganic structures

As there were far fewer inorganic than organic structures in CrystalEye, the aim was to select all of them prior to filtering and calculation. The filtering

steps are summarized below:

1. select all inorganic structures from CrystalEye — giving 17919 structures,
2. remove those structures containing disorder — leaving 7404 structures,
3. remove structures that contain pairs of atoms for which MOPAC has no atom-pair parameters — leaving 2245 structures,
4. remove structures that do not contain an author provided formula — leaving 2173 structures.

It is worth noting that CrystalEye contains many ‘duplicate’ structures which have the same composition, but are either different phases, or are the same structure but the experiment was performed at a different temperature or pressure. All duplicate structures that passed the above filters were retained in the dataset, and shall be further discussed later.

From these 2173 structures, 500 were selected and the calculations performed as described previously. However, upon analysing the output, it appeared that the results were unpromising and so no further calculations were performed at that time. Of the 500 calculations:

- 464 converged successfully,
- 5 reached the time limit before converging,
- 3 reached the cycle limit before converging,
- 28 resulted in MOPAC errors that gave controlled abortions,
- 0 resulted in an uncontrolled abortion.

For those calculations that converged successfully, figure 5.10 shows a plot of the experimentally observed crystal density against the change in density predicted by MOPAC. The average unsigned error in calculated density for this dataset is 25.1%, which is larger than is acceptable.

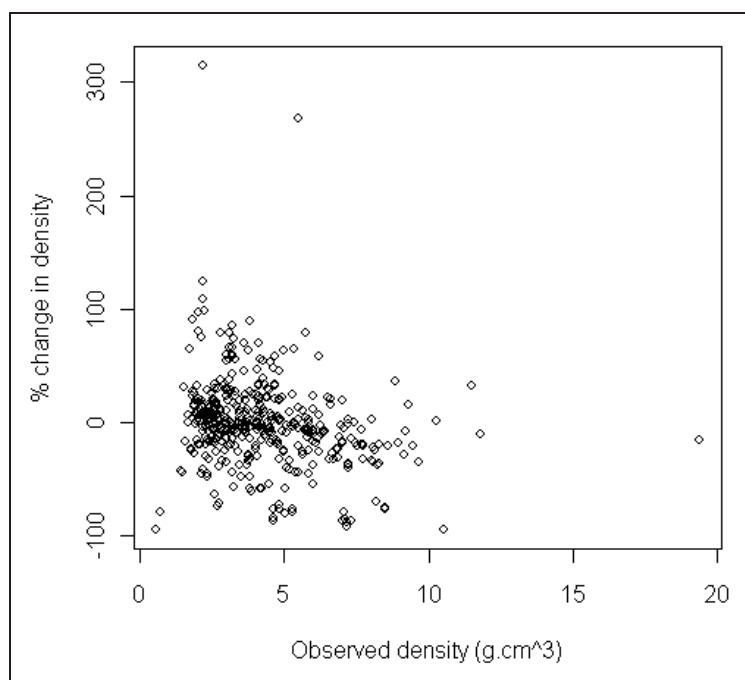


Figure 5.10: Plot showing the experimentally observed densities against the change in density predicted by MOPAC for the set of 464 successfully converged inorganic structure calculations.

After this finding, I decided that rather than continue with the analysis of the inorganic calculations, I would wait for the next development in the protocol for solid-state calculations. I also decided that the full analysis of the organic calculations would be postponed to this time. It was at this point that the work on Open Computational NMR was performed (see chapter 6).

## 5.6 The second protocol

In March 2008, Stewart published a second paper on PM6[179], which described his calculations performed on the solid-state. In this paper, the results of geometry optimization for 124 organic structures and 166 inorganic structures are discussed, with the starting structures for each calculation being the observed X-ray structure. The reference data for the organic and inorganic structures were obtained solely from the Cambridge

Structural Database[132, 133] and American Mineralogist Crystal Structure Database[195].

The protocol used by Stewart for the organic calculations was the same as was used in my work. With few exceptions, the geometries of individual organic molecules and ions and their packing arrangement were reproduced with good accuracy. The average unsigned and signed errors for the calculated density of organic solids was 6.9% and 3.9% respectively. This matched well with values obtained during this work of 6.4% and 4.0%. Stewart attributed most of these errors to problems in modeling intermolecular separation. No major errors were found during these calculations, though Stewart did note that formation of the  $\text{H}_3\text{O}^+$  ion was unrealistically favoured.

The parameters used by Stewart for the inorganic calculations were the same as I had used, though the minimum cluster size differed, with Stewart stating that:

“... a much larger cluster must be used when inorganic solids that are not composed of discrete molecules are modeled. In practice, this means that the cluster used has to be large enough to contain a sphere of radius 10-12Å, in contrast to the 7-8Å used in the modeling of organic solids.”[179]

The average unsigned and signed errors of calculated density for the inorganic calculations were stated as 9.3% and -2.5% respectively. The unsigned error is a large improvement on the 25.1% observed during the calculations, and signified that I should rerun them, and the remainder of the dataset, with the larger cluster size recommended.

Most of the inorganic structures used by Stewart contained common, well characterized anions such as halides, oxides, carbonates, phosphates, sulfates, sulfides, nitrates and silicates, which may have been the reason for the improvement in accuracy. Barring silicates, the majority of the structures were of type  $\text{A}_n\text{B}_m$ , or contained two distinct species. The highest accuracy

was exhibited in minerals in which metal atoms interact with oxygen, with the oxygen then interacting with a main group element. Stewart attributed this to the fact that a large quantity of reference data for systems that have such interactions was used during parameterization.

During the work, Stewart found five atom-pairs where the core-core parameters were erroneous, and which:

“...gave rise to results that were nonsense.”[179]

These erroneous pairings were Na-Na, Pb-Sb, Pb-Zn, Pb-Se and Br-N. Stewart stated that fixing these problems:

“...would involve only a re-optimization of the faulty diatomic parameters, and would not alter the performance of PM6 when applied to other systems.”[179]

Note that a new version of MOPAC had not been released since these findings, so any occurrences of these atom-pairs in the calculations was likely to also give poor results. As Stewart had used a much smaller range of systems than were available in the selection from CrystalEye, it was likely that I would uncover further erroneous atom-pairings, and so it seemed sensible to investigate this early in the analysis.

Before any calculations were run with structures from CrystalEye, I repeated all of those calculations that Stewart had performed and made available via the MOPAC website[196]. The results obtained from the calculation environment used in this work agreed with those obtained by Stewart.

### 5.6.1 The data

The MOPAC input files (`.mop`), the standard MOPAC output (`.out`), the MOPAC-CML output (`.xml`) and the original CIF from which the input files were created (`.cif`) for all organic and inorganic calculations have been made available through the University of Cambridge’s DSpace repository at the following addresses:

- <http://www.dspace.cam.ac.uk/handle/1810/197582>
- <http://www.dspace.cam.ac.uk/handle/1810/197581>

For each of the two datasets, the names of the files have the form of **ned24-xx**, where **xx** is an integer that refers to the job number of the calculation. When discussing groups of calculations in the following analyses, I will refer to sections in appendix B, which will contain a list of the names of the relevant jobs.

### 5.6.2 Inorganic structures

From the 2173 structures selected in section 5.5.2, MOPAC input files were regenerated with the same input parameters, though with a larger minimum cluster size of 12Å, the upper limit of the minimum range recommended by Stewart. No extra input parameters were used to specify the spin-state of any metal centres, though as Stewart has stated:

“For geometries, the effect of ignoring spin is minor — much less than the errors in PM6.” [197]

this should not effect any conclusions that I draw from the results.

From the created MOPAC input files, all those jobs containing clusters with 300 or fewer atoms were selected, giving 1731 structures from the following sources:

- Acta Crystallographica - 172 structures
- American Chemical Society - 52 structures
- Royal Society of Chemistry - 14 structures
- Crystallography Open Database - 1493 structures

From the calculations performed on these structures:

- 1258 converged successfully,

- 424 reached the time limit before converging,
- 1 reached the cycle limit before converging,
- 48 resulted in MOPAC errors that gave controlled abortions,
- 0 resulted in an uncontrolled abortion.

Figure 5.11 shows a plot of the experimentally observed crystal density against the change in density predicted by MOPAC for those calculations that converged successfully. The average unsigned and signed errors in calculated density for this dataset are 17.8% and -4.8% respectively. The unsigned error is considerably higher than that observed by Stewart in his work, though it is perhaps expected, as a far wider range of inorganic systems have been calculated in the dataset in this work. It is likely that some of the more exotic atom pairings would have had a poor representation during parameterization, which can lead to incorrect parameters, and hence results.

Those calculations that either ran out of time or reached the cycle limit before converging will not be rerun, and hence will not figure in any of the further analysis. The calculations that terminated with controlled errors are discussed at the end of this subsection.

### Short atom-atom distances

Erroneous atom-pair parameters can be discovered by inspecting the final geometries of each calculation for particularly short atom-atom distances. A program was written to iterate through all of the MOPAC-CML files for the calculations that converged successfully to find any instances where the atom-atom distances were less than three-quarters of the sum of the covalent radii.

This program found 24 different atom-pairings with short atom-atom distances in 57 structures; these are summarized in appendix B.1.1. The atom pairs are Ba-Ba, Br-Ta, Ca-Na (as shown in figure 5.12), Ca-S, Cd-N, Cl-Fe, Cl-Hg, Cs-P, F-K, Hg-Hg, Hg-Te, K-Nb, K-Ta, La-La, Mg-Mg, Na-Na, Na-S,

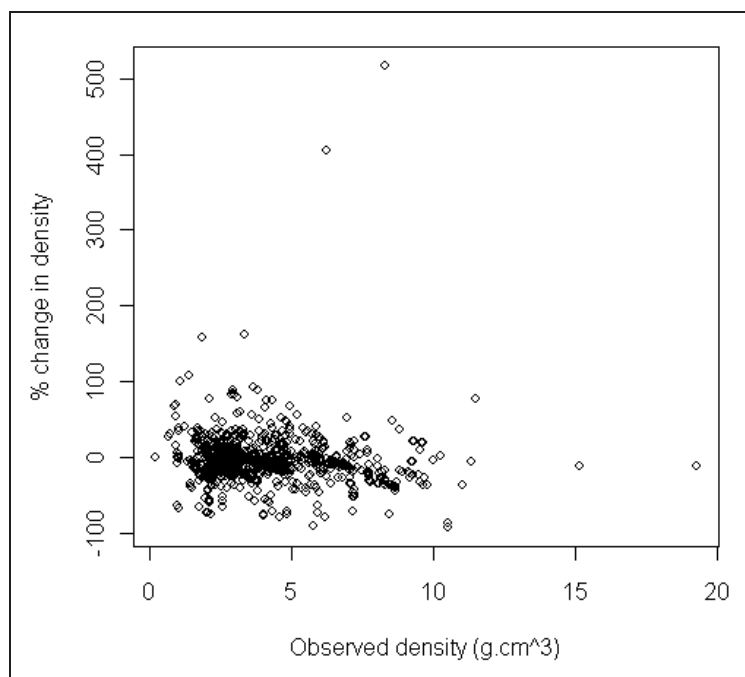


Figure 5.11: Plot showing the experimentally observed densities against the change in density predicted by MOPAC for the set of 1258 successfully converged inorganic structure calculations.



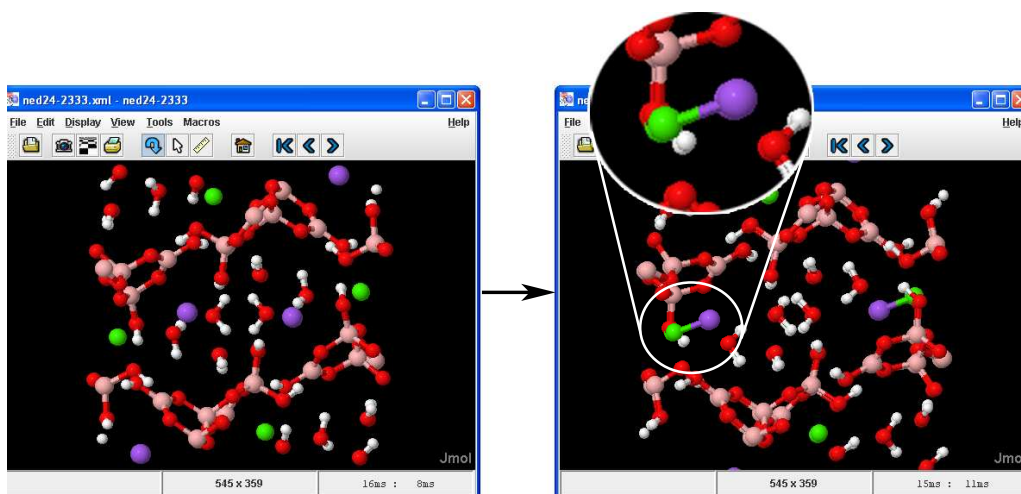


Figure 5.12: Figure showing the creation of a spurious Ca-Na bond in prober-tite. The Ca-Na distances in the starting and final structure are  $3.8\text{\AA}$  and  $1.9\text{\AA}$  respectively (green=Ca, purple=Na).

Na-Ta, P-Pb, Pb-Pb, Pb-Se, Si-Sr, Zn-Zn. I inspected the bond-length plots at CrystalEye[198] corresponding to each of these atom-pairs, and in each case the calculated distance was less than that of all observed distances in the entire CrystalEye dataset. 496 structures containing one or more of these atom-pairs were found in the dataset, and were subsequently removed from further analysis. After this filtering, 762 structures remained.

### Duplicate structures

As discussed before, no attempt was made to remove structures with duplicated composition from the dataset. Of the remaining 762 structures, 542 had structures of duplicate composition. These duplicates may consist of different phases of the same structure or structures of the same phase that had been elucidated at a different temperature and/or pressure.

By investigating the results of those calculations for structures of identical composition and phase at different temperature and pressure, it is shown that they converge on the same structure. This is not an unexpected result, as all calculations are performed at absolute zero, though it is gratifying.

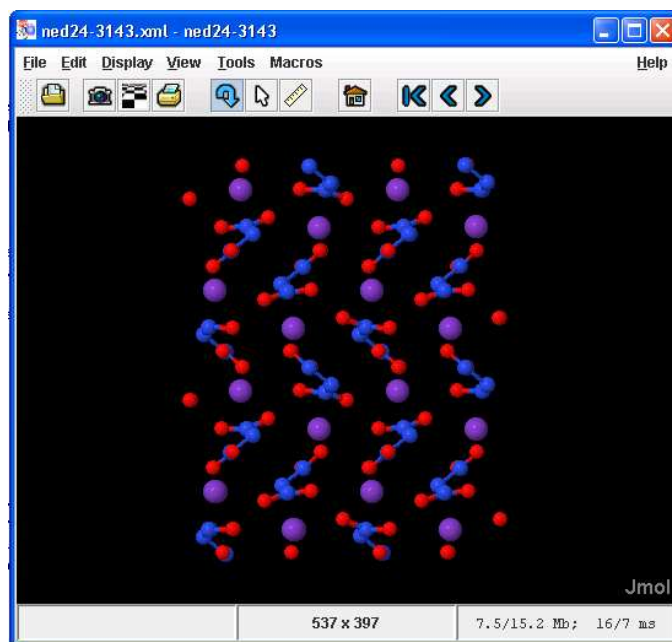


Figure 5.13: Figure showing the starting cluster of potassium dinitramide displayed in Jmol (where purple=K, blue=N, red=O)

It does, however, highlight a source of error in the difference between the observed and calculated density. For instance, in the dataset, there were 7 structures of potassium dinitramide,  $\text{KN}_3\text{O}_4$  (as shown in figure 5.13) at temperatures between 85K and 298K. There is a 3.7% difference in starting density between the minimum and maximum of these structures; therefore, if the temperature of the structure of potassium dinitramide that is being calculated is not known, then an error of less than 3.7% cannot be expected. This order of error can be expected for other systems, and so it may be useful to analyse only those structures that were observed at low temperatures and pressures. However, this metadata is not routinely provided for all structures in the dataset, and due to the already small number of structures left, I decided not to perform this filtering.

From the 542 structures with duplicate composition, each was examined and one representative of each phase of each structure was selected using the

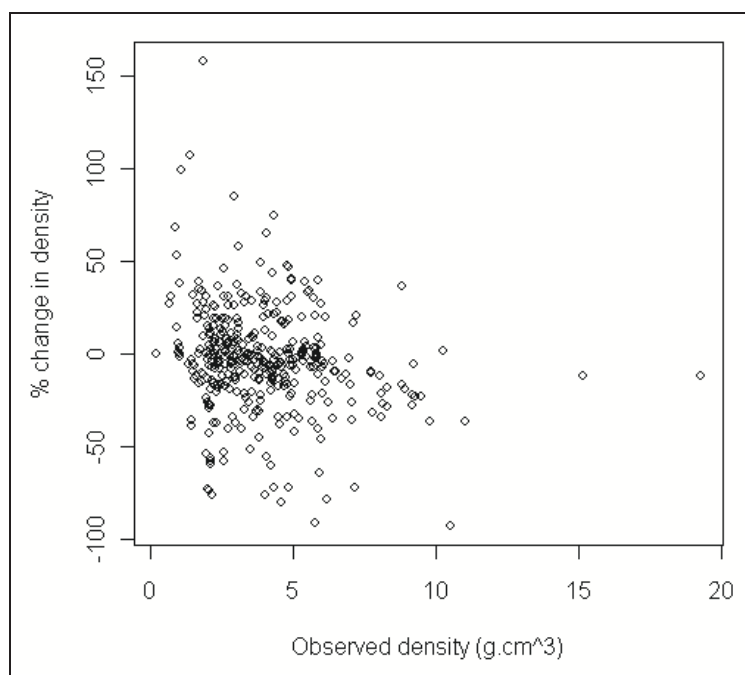


Figure 5.14: Plot showing the experimentally observed densities against the change in density predicted by MOPAC for the set of 393 remaining inorganic structure calculations.

following criteria:

- where possible, select the structure with the lowest temperature or pressure (there were no instances where both were provided),
- if no temperature or pressure is provided, select the structure with the smallest density change during calculation.

173 structures were selected during this filtering. The other 369 duplicates were removed from the dataset, leaving 393 structures.

For the 393 calculations left in the dataset, the plot of the experimentally observed crystal density against the change in density predicted by MOPAC is shown in figure 5.14. The average unsigned and signed errors for these structures are 19.5% and -3.5% respectively.

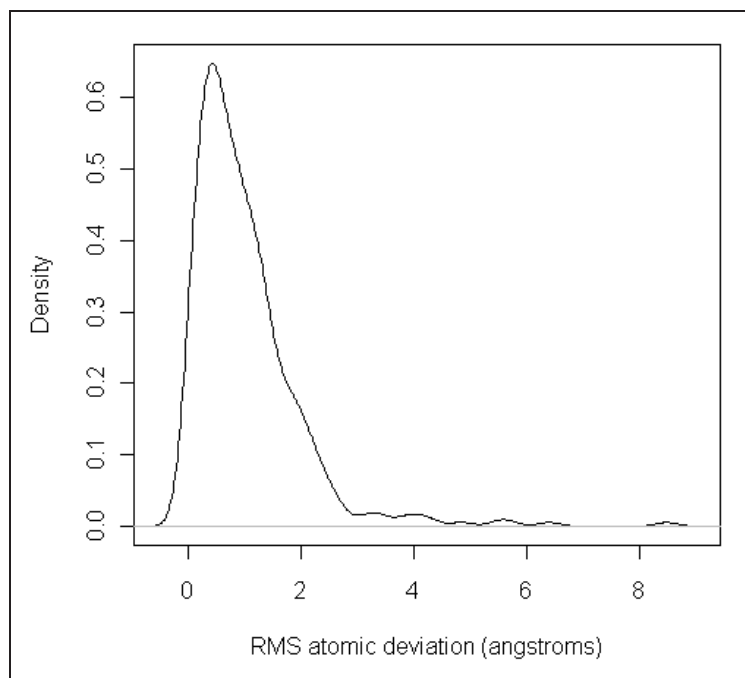


Figure 5.15: Density plot of the RMS atomic deviations for the 393 remaining calculations. Note that the appearance of RMS values of  $< 0\text{\AA}$  is an artifact of the density kernel method in R[201].

### Atom RMS deviations

Inspecting change in density allows the discovery of poorly modeled structures where there has been a large expansion or contraction. However, to find those structures where there have been large atomic rearrangements but the density remains unchanged, another method is needed. This can be achieved by using the RMS atomic deviation during calculation. For each of the remaining 393 calculations, the density plot of the RMS deviations are shown in figure 5.15, where the mean RMS deviation is  $1.0\text{\AA}$ .

Starting with the structures with the largest RMS deviation and working downward, the observed and calculated structures were inspected for each calculation. The aim was to find the range in which those structures that had been calculated poorly fell. It appears that for those structures with an RMS deviation of less than  $1\text{\AA}$ , the overall structure remains largely unchanged.

The changes in this area involve small expansion/contraction or rotation of subspecies. There are 242 such structures, which is 62% of the dataset. In this range, there is a preponderance of ionic structures of light and medium elements, and a small number of compounds that contain transition and semi-metals. The latter group generally contain well known solvents, the lighter halogens, or metals that are bonded only to oxygen atoms. These types of structures that I observe to be modeled well by MOPAC are in agreement with Stewart’s work in the second PM6 paper[179].

For structures with an RMS deviation greater than 1Å, there are still many examples where the change involves expansion, contraction or rotation, though each is naturally more pronounced as the RMS deviation increases. As the maximum RMS deviation is approached, there are increasing numbers of structures where large atomic rearrangements are observed, sometimes with a loss of symmetry. In this range, there are many structures containing the heavier halogens, the heavier alkali and earth-metals, and transition and semi-metals bound to elements other than oxygen. For these structures, the cause of the large RMS deviations can be assigned to three categories:

1. erroneous starting data,
2. real chemical variability,
3. modeling problems in MOPAC.

### **Erroneous starting data**

10 calculations were found where the starting structure appeared to be incorrect (see appendix B.1.3), owing to the following problems:

1. missing hydrogen atoms — these structures were not removed during the filtering as the formula provided in the CIF did not contain the H atoms either.
2. incorrect symmetry elements leading to the generation of an incorrect unit cell — for these structures, the symmetry elements had not been

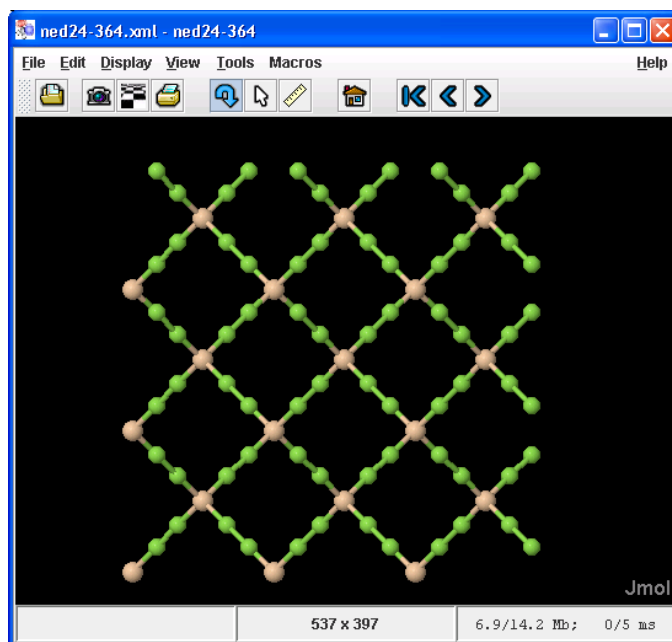


Figure 5.16: Figure showing the erroneous starting structure of  $\text{SiF}_4$  containing F-F bonds.

provided in the CIF, but had been added during processing by CrystalEye. In each case, the wrong space group had been provided in the CIF, leading to the addition of the wrong symmetry elements during the CML creation.

3. incorrect data provided in the CIF, *e.g.* the structure of  $\text{SiF}_4$  shown in figure 5.16, which contains F-F bonds.

As 233 of the 393 structures were inspected, this gives a 4.3% error rate, which is far higher than desirable. 8 of the 10 erroneous structures came from the COD, though this is not unexpected, as over 86% of the calculated structures were from this source.

### Chemical variability

Some structures are known to be very difficult to model as they undergo large distortions for very little energy. For instance, eight of the phases of silica are

present in the dataset (see appendix B.1.2<sup>†</sup>). These were low-quartz, high-quartz, low-tridymite, high-tridymite, low-cristobalite, high-cristobalite, coesite and stishovite. The structural rearrangements between the high and low form of each phase involve low energy rotation of tetrahedra, with transition between phases involving changes in the linkage between the tetrahedra. In each case, as expected, the linkage between the SiO<sub>4</sub> tetrahedra remains the same during calculation, and for high-quartz, high-tridymite, high-cristobalite, coesite and stishovite, no tetrahedral rotation is predicted. However, for low-quartz, low-tridymite and low-cristobalite, MOPAC predicts that they will undergo a phase transition into the higher symmetry, higher temperature forms. This is shown in figure 5.4, where low-cristobalite is rearranged during calculation into high-cristobalite.

There are other structures which are observed to undergo phase changes during calculation (which may or may not be real), though investigating them all is beyond the scope of this thesis.

## Problems in modeling

The observed problems in modeling fall into three categories, these are:

- uncontrolled expansion,
- large loss of symmetry,
- formation of new bonds.

The job name relevant to each of the points discussed below are provided in appendix B.1.4. Only one example of uncontrolled expansion was found in the dataset, which was for the structure of NiF<sub>2</sub> (see figure 5.17).

MOPAC does not contain any specific functionality to handle symmetry, and so for those interactions that are poorly modeled, this can lead to a large loss of symmetry. This is usually seen when bonds are formed or broken, and

---

<sup>†</sup>note that there were 106 structures of SiO<sub>2</sub> in the dataset before duplicates were removed

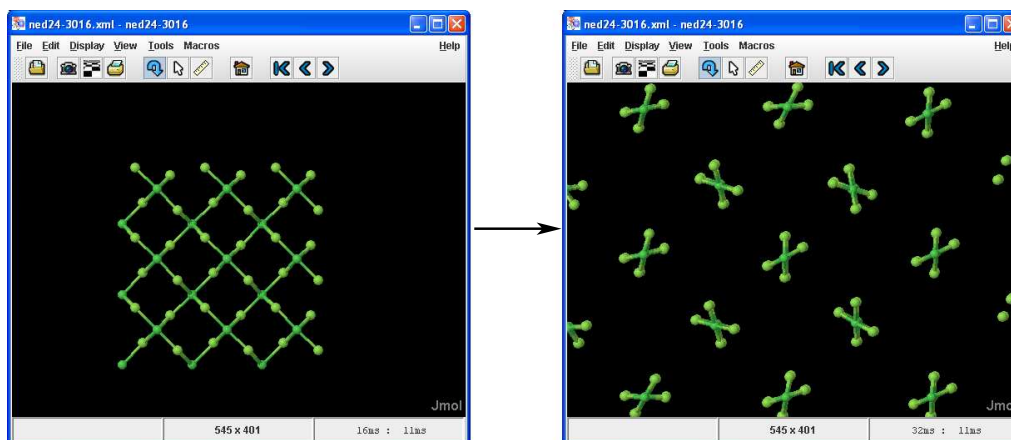


Figure 5.17: Figure showing the large expansion in geometry predicted for  $\text{NiF}_2$ .

was noted in 20 cases. For instance, three of the structures with the highest RMS deviations contain Ag, where a large loss of symmetry is predicted in each case. This is shown in the calculation for  $\text{Ag}_2\text{O}$  in figure 5.18. it is interesting to note the well-formed structure that breaks away from one corner of the cluster. The elements that occur more than once in structures that show a large loss of symmetry are (number of occurrences shown in parentheses) Ag(3), Rb(3), Cu(2), Xe(2) and Ga(2).

During inspection of the calculations, 4 atom-pairs were noted on more than one occasion to give rise to bond formation<sup>‡</sup>, I-I, Rb-F, Al-Si and Tl-O.

### Calculations that terminated with controlled errors

The 48 calculations that failed with errors did so with two different types:

- unable to achieve self-consistency - 6 occurrences
- numerical problems in bracketing lambda - 42 occurrences.

Failing with these errors can signal problems with the modeling in MOPAC, thus I should have removed other jobs containing these pairs from the analysis

<sup>‡</sup>where a bond is taken as a distance of less than 1.2 times the sum of the covalent radii



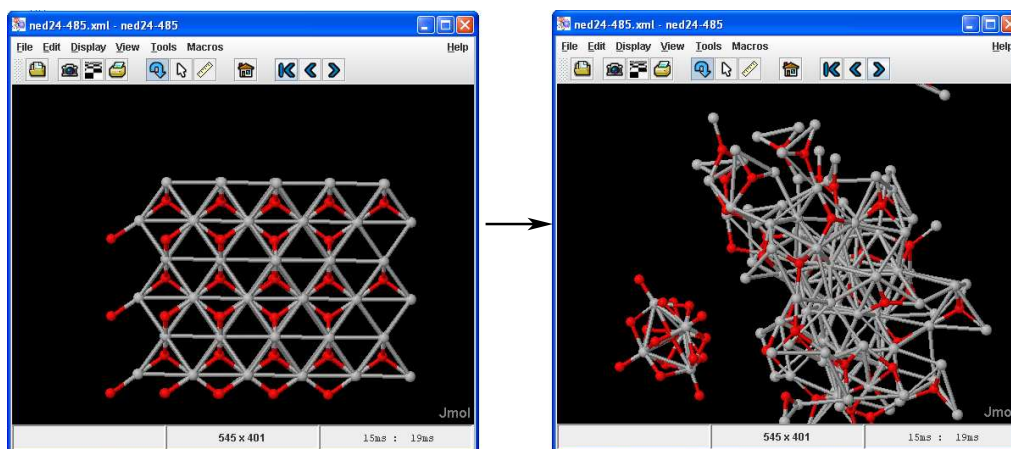


Figure 5.18: Figure showing the large loss of symmetry predicted during the calculation for  $\text{Ag}_2\text{O}$ .

at an earlier point. Details for all jobs discussed in the points below are provided in appendix B.1.4.

There are 14 elemental crystal structures included in these failed jobs, Li (2 structures of the same phase), Sc, Sr, Cd, Au, Tl (3 structures of two different phases), Pb and Bi (4 structures of the same phase). Except for Li, in each case the symmetry is retained, though there are large expansions before the calculation terminates. The structures for Li show a large loss of symmetry, with some bond lengths being predicted to be less than  $1.4\text{\AA}$ .

### 5.6.3 Organic structures

As the initial set of organic calculations were performed with the same protocol Stewart used in the second PM6 paper, I can continue the analysis where I left off in section 5.5.1.

Again, as with the analysis of inorganic calculations, the 1526 calculations that either reached the time limit, or the maximum number of cycles before converging will not be rerun, and will not be discussed further in this analysis.

## Calculations that terminated with controlled errors

Two different errors were observed in the 27 failed calculations (summarized in appendix B.2.1):

- `numerical problems in bracketing lambda` — 8 occurrences.
- `all convergers are now forced on` — 19 occurrences.

For the former, it was noted that all of the structures contained a nitroxyl radical. When calculating the geometry of radicals, the `UHF` keyword should be used, but all of the calculations in this work were performed using the default `RHF`. Thus, the results for any calculations containing radicals must be removed from the analysis. A simple program was created to iterate through the dataset to find all structures containing moieties with an odd number of electrons and remove them. During this filtering 8 calculations that converged successfully were removed (summarized in appendix B.2.2), leaving 4237 remaining.

For the latter error type, the online MOPAC manual states that:

“This is often caused by faulty data, so the data should be checked to see if anything is wrong. This sometimes happens naturally, particularly with exotic systems.” [199]

Each of the 19 structures causing this error were inspected. The starting structure of one calculation (`ned24-8752`) was found to be erroneous, as the position of one H atom in the original CIF was incorrect (see figure 5.19). The other 18 starting structures were found to be correct and are described in table B.1 in appendix B.2.1. On inspection, it appears that the following may give problems during calculation in MOPAC:

- structures containing a high proportion of N or F atoms,
- structures containing N-N bonds,
- structures containing C bonded to two or more N atoms.

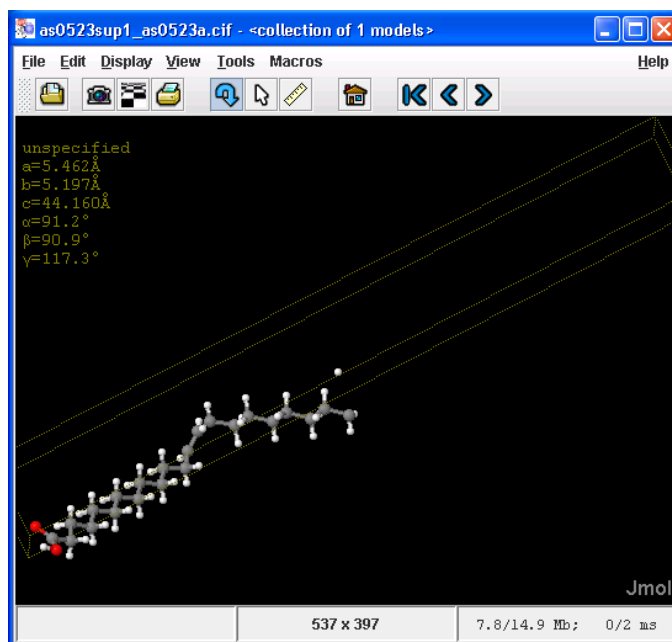


Figure 5.19: Figure showing the data in the CIF for erucic acid rendered in Jmol. Note the incorrect position of one of the terminal H atoms.

Townsend's work has previously highlighted 10 molecular fragments that seem to give problems when performing geometry optimization in the gas-phase using MOPAC (though he was using the PM3 method)[200]. 9 of those 10 fragments contained either N-N bonds or C bonded to two or more N atoms.

### Changes in connection table

By inspecting the connection tables of the moieties in the cluster throughout each calculation, the formation or breaking of any bonds can be observed. A change in connection table during calculation can be caused by two things:

- an erroneous starting structure containing 'wrong' bonds,
- incorrect modeling of the system by MOPAC.

A program was written to iterate through the MOPAC-CML for each calculation, and to generate the InChIs for the starting, intermediate and final

structures. It then compared the InChIs at each stage and noted each calculation where the InChI did not match. The job names relevant to each point discussed below are provided in appendix B.2.3. Stewart has previously discussed the unrealistic favouring of Zwitterion formation from amine and alcohol groups, as well as formation of the oxonium ion when using PM6[179]. As a result, these types of H-shift will not be further investigated here.

The program highlighted 87 structures which had undergone a change in connection table during calculation. Of these, 85 contained bond formation and only 2 contained bond breaking. Upon inspection it was found that these 2 structures had incorrect starting geometry, which was traced back to the fact that the CIFs for each (**ned24-9025** and **ned24-10389**) did not contain any symmetry elements. As with the inorganic examples earlier, when these CIFs were processed, CrystalEye had used the provided space group to add symmetry elements to the data. In each case this space group was incorrect and meant the wrong symmetry elements were added for the cell, leading to the creation of a structure which contained close contacts. In the case of **ned24-9025**, these consisted of short C-H bonds, and in **ned24-10389** there are also short C-I bonds. In both cases, MOPAC 'fixes' the broken structures.

For the other 85 structures, only bond formation is observed<sup>§</sup>, leading to the creation of charge-transfer complexes. Most of the bonds formed are between p-block elements, with the most common involving the heavier halogens. The elements involved and the number of structures in which they were observed to bond are:

S-N (3); S-O (4); S-S (2); S-Cl (2); Se-Se (2), Se-I (4), Te-Cl (1),  
O-Br (4), N-Br (12); Br-Br (2); O-I (38); S-I (3); N-I (4); I-I (7)

The other observed effect was in hydrogen bonds containing the chloride ion. 13 structures were found where there is a significant shortening of the A-

---

<sup>§</sup>where a bond is taken as a distance of less than 1.2 times the sum of the covalent radii

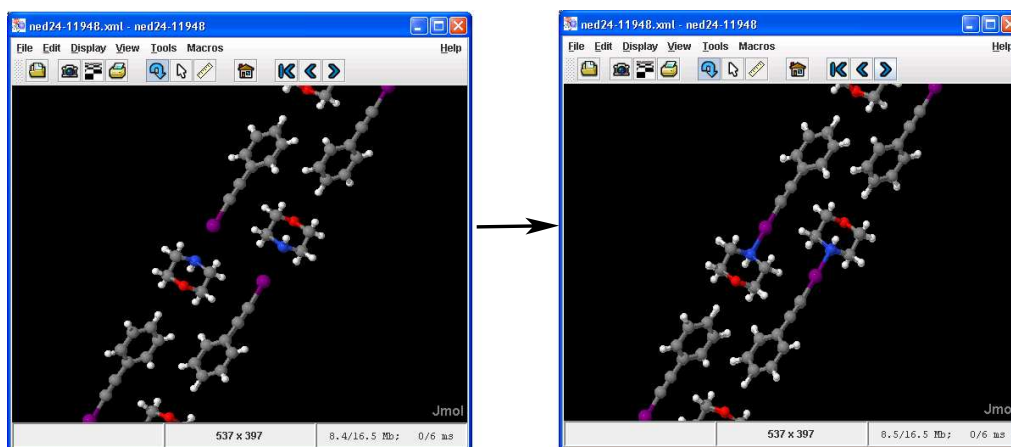


Figure 5.20: Figure showing the formation of an N-I bond by contraction of the cell. The observed N-I distance is  $2.7\text{\AA}$ , while the calculated distance is  $2.1\text{\AA}$ .

H—Cl distance, with the H atom also moves noticeably towards the A-Cl midpoint.

On inspection of the structures where two p-block elements are shown to bond, as expected, the new bonds are generally intermolecular, and involve intermolecular shifts, rather than intramolecular movement to bring the atoms closer together. It appears that where the atom-pairs are not obstructed by other species, it is normally a simple contraction of the cell that leads to bond formation, as shown in figure 5.20.

The only cases of intramolecular bond formation found are for S-S, and in both cases, this is brought about by an intramolecular 1,5 interaction, as shown in figure 5.21.

In the other 83 cases where bonds are formed, the new bond is intermolecular. The effects observed indicate that MOPAC seems to overestimate the strength of charge-transfer bonds (see figure 5.20). In each case where S-N bonds are formed, either the N or S atom is already bonded to another N atom (as shown in figure 5.22). In the structures where S-O bonds are

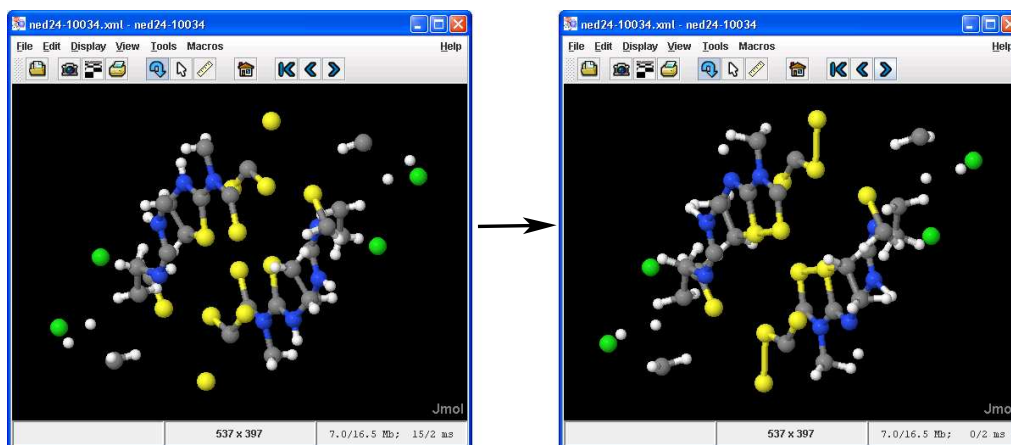


Figure 5.21: Figure showing the formation of an S-S bond (yellow=S, blue=N, green=Cl). The S-S distances in the starting and final structures are 2.9Å and 2.3Å respectively.

formed, the oxygen is either part of a carboxylate or nitro group (there are two examples of each).

In an attempt to ascertain the distances at which these atom-pairs become attracted to each other, a program was written to iterate through all the calculations, and for each atom pair described above, plot all distances of less than 5Å in the starting structures against the distance between the two atoms in the final structure. The plot for O-I interactions is shown in figure 5.23. There is clearly a distance around 2Å favoured for O-I in many calculated structures.

This same plot pattern is observed for many of the other atom-pair combinations described above, where the majority of the interactions lie around the line of unity, though few show a large decrease in atom-atom distance. While in the plot for O-I it appears that a distance of over 4.3Å is enough to stop O-I formation, in the plot for N-Br (figure 5.24), it appears that distances of over 5Å can still lead to bond formation. Again, as described above, those interactions where the distance is considerably reduced during calculation seem to arise in structures where a cell contraction can be made

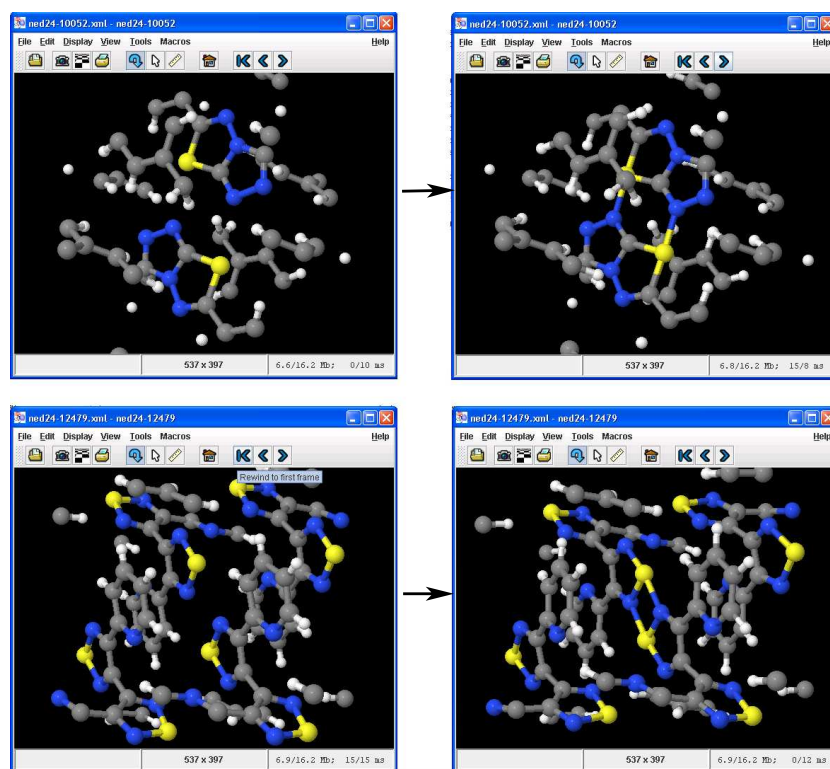


Figure 5.22: Figure showing two structures in which the formation of a new S-N bond is predicted (yellow=S, blue=N).

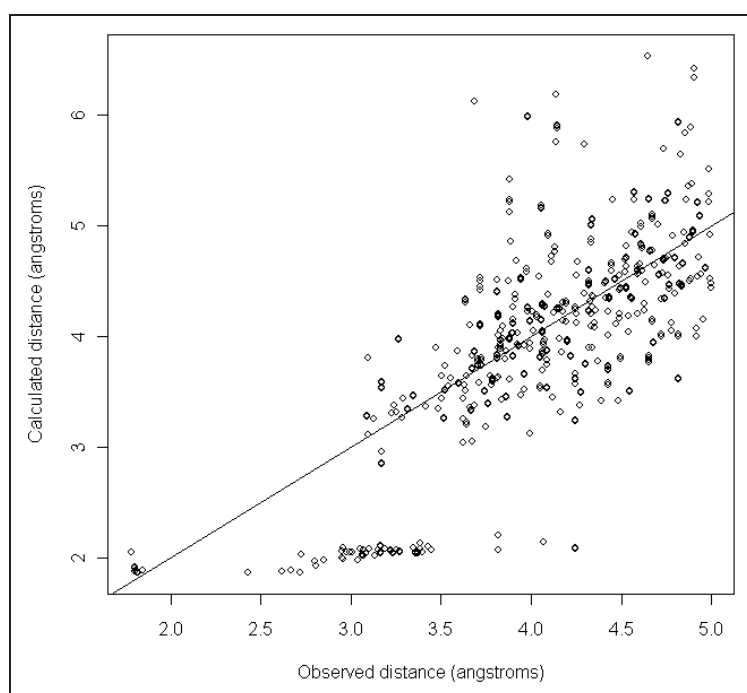


Figure 5.23: Plot of all observed O-I distances of less than  $5\text{\AA}$  against the calculated distance (the line shown is unity).



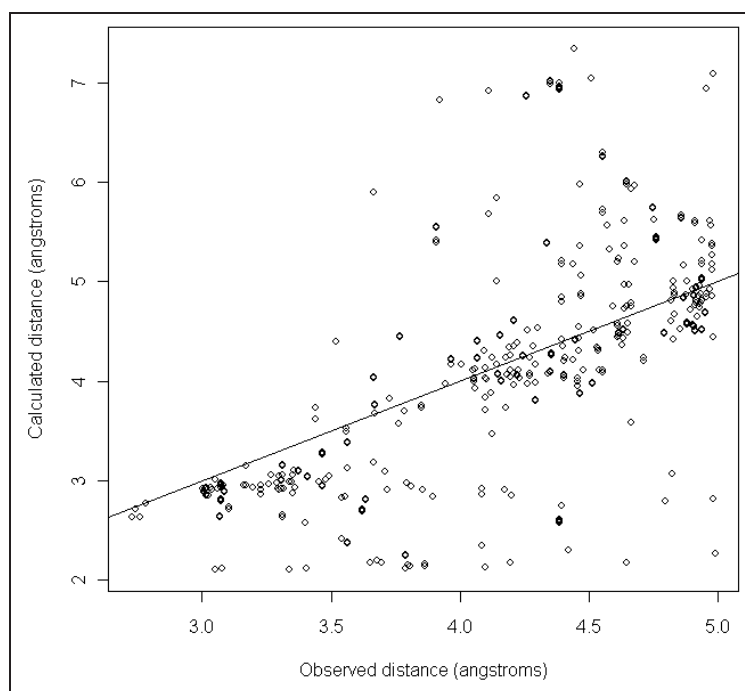


Figure 5.24: Plot of all observed N-Br distances of less than 5Å against the calculated distance (the line shown is unity).

to bring atoms in different moieties closer together.

### Atom RMS deviations

For the 4237 remaining structures that converged successfully, the density plot for the RMS atomic deviations is shown in figure 5.25. The mean RMS deviation is 0.77Å, higher than expected, and there were more than 50 structures with RMS deviations of over 5Å. After investigation, it was found that in many cases, the RMS deviations were artificially high due to the entire cluster undergoing rotation, even when the internal structure of the cluster remained unchanged (as shown in figure 5.26). This was a surprising effect that neither I nor Stewart had expected. The origin of this problem is under investigation by Stewart and has yet to be confirmed.

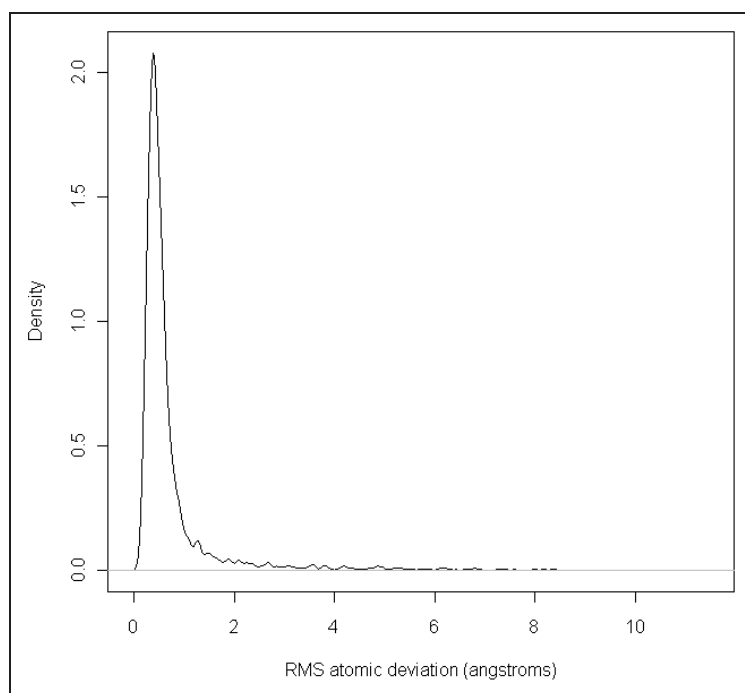


Figure 5.25: Density plot of the RMS atomic deviations for the 4237 remaining calculations.

A program was then written to ‘fit’ the final structure back on to the starting structure and identify the magnitude of the rotation that had occurred. Figure 5.27 shows the plot of structural rotation against the RMS deviation for that structure. This rotation of the entire structure makes it more difficult to write software to identify patterns in the calculations. As such, further analysis of RMS deviations will be postponed until the issue with cluster rotation has been investigated.

### Minimum supercell size

Owing to the nature of the solid-state calculations in MOPAC, the size of the cluster used can have an effect on the accuracy of the calculation. As discussed above, the minimum translation vector during the organic calculations was  $7\text{\AA}$ . For unit cells with lengths less than  $7\text{\AA}$ , a supercell must be created. This means that for most calculations, the smallest translation vec-

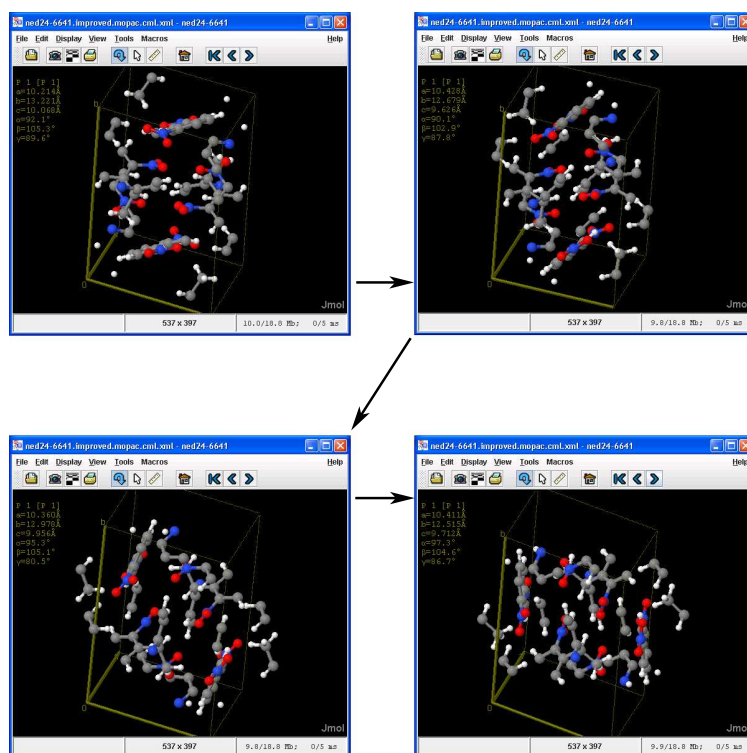


Figure 5.26: Figure showing the rotating geometry of  $\alpha$ -(2-pyridine)-2,4-dinitrophenylethynyl at various stages during calculation.

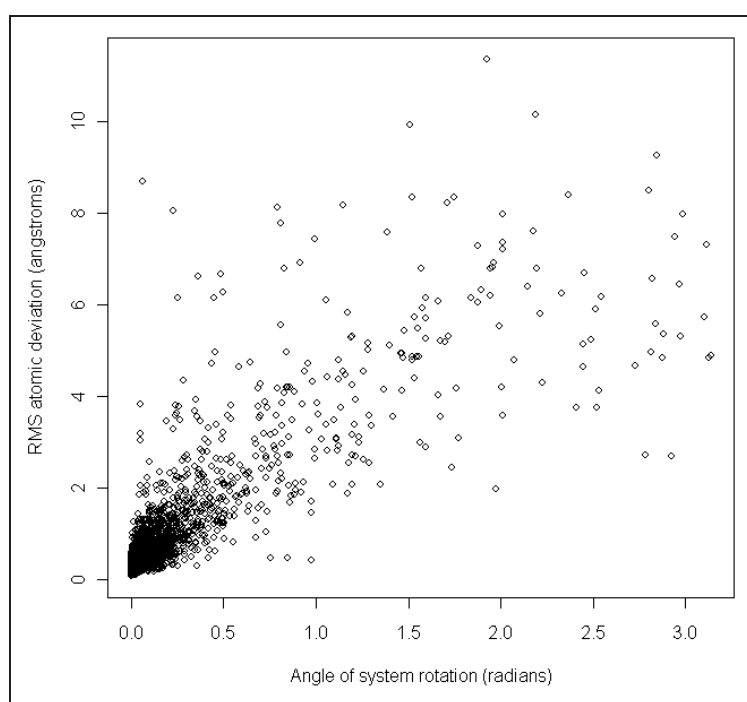


Figure 5.27: Plot of the angle of rotation of each structure against its RMS deviation.

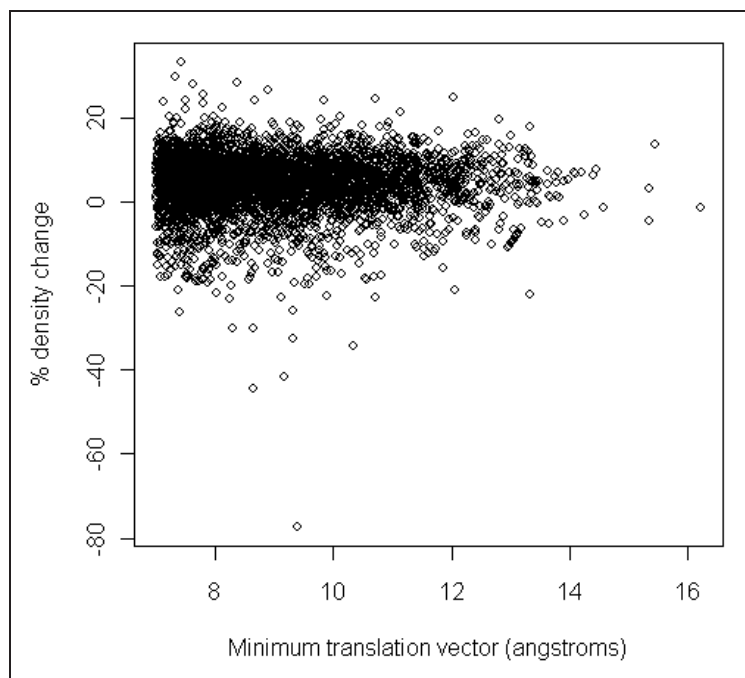


Figure 5.28: Plot of the minimum translation vector against the change in density during calculation for each structure.

tor will lie between  $7\text{\AA}$  and  $14\text{\AA}$ . Thus, by plotting the minimum translation vector against the density change, I can get an idea of the improvement in accuracy afforded by the larger clusters (as shown in figure 5.28).

It is not clear just by inspecting the plot whether there is a relationship between the two. To do this, the structures can be split up into ranges of minimum translation vector, and then calculate the standard deviation and mean for each (table 5.6.3). There does not seem to be an improvement in either the mean or standard deviation of density change as the minimum translation vector increases. Thus when calculating a large number of organic structures containing a range of systems, there appears to be no benefit from using a minimum translation vector greater than  $7\text{\AA}$ .

Min. TV (Å)	No. of structures	Mean	Standard deviation
$7 < x \leq 7.5$	714	4.06	6.75
$7.5 < x \leq 8$	685	4.00	7.21
$8 < x \leq 8.5$	542	4.25	6.70
$8.5 < x \leq 9$	446	4.09	6.82
$9 < x \leq 9.5$	418	3.22	8.00
$9.5 < x \leq 10$	374	3.73	6.74
$10 < x \leq 11$	548	4.21	6.38
$x > 11$	516	4.13	6.33

Table 5.1: Table showing the density change mean and standard deviation for structures with various minimum translation vectors.

### Density change outliers

To find those structural fragments that commonly give rise to large changes in density during calculation, all those structures that had a predicted absolute density change of  $>20\%$  and did not contain a change in connection table were inspected. For density changes of  $<-20\%$  there were 16 structures, while for changes of  $>20\%$  there were 13 structures. These are shown in tables B.2 and B.3 in appendix B.2.4. As observed in earlier examples, the change in density for each of the inspected structures can be attributed almost exclusively to changes in intermolecular distances.

For those structures with predicted density change of  $<-20\%$ , there are three structural features which are prominent:

- structures with a high proportion of F or Cl atoms, or containing a centre bonded to 3 F or Cl atoms — 7 examples,
- structures containing a dithiocyclopentene fragment — 5 examples,
- structures containing aromatic N — 5 examples.

There is also one structure (ned24-7257) which contains a C atom bonded to 3 N atoms, which was a fragment highlighted earlier that occurred in those calculations which terminated with controlled errors.

For those structures with predicted density change of  $>20\%$ , there are also 2 structural features which stand out:

- aromatic groups with Cl, Br or I substituents — 5 examples,
- icosahedral carboranes — 2 examples.

## 5.7 Conclusions

The high-throughput calculation of several thousand solid-state structures has been used to highlight and report several issues that need to be addressed in the algorithms and parameterization of the PM6 method. Stewart has stated:

“Most of the errors were either new to me, or were good examples of known faults.” [203]

Creating the workflow and performing these high-throughput calculations would have been impossible if the data had not been Open, and would have been considerably harder if CrystalEye had not been created to provide the data in a readily available XML-based collection.

By running several thousand jobs with a fixed set of parameters, I was able to discover many issues with the method. The MOPAC algorithm for solid-state calculation, appears to be robust, *i.e.* no uncontrolled crashes were encountered during the 8029 calculations performed in this work. It is important to ‘stress-test’ methods in this way, as the more systems that are calculated, the more problems are discovered. For semi-empirical methods this is particularly important, as the data that gives rise to errors can often be used in the next parameterization to fix the problem. Stewart has stated that:

“Developing tests for new methods is hard work, and until now has been ‘hit or miss’ — mostly I wait until someone writes in with a complaint, and most of those are user errors anyway, and

thus result in a lot of wasted time. What you’ve done is, I think, on a par with the CSP2004 test[202], but with a very different emphasis. In CSP2004, the objective was to use skill to predict the packing of new solids. Once the test was complete, it could not be used again. In your work, you provide a test that can be applied to any new computational chemistry method, by the authors of that method, or by users.”[203]

The calculated results for organic structures, generally agree well with experiment. Several atom-pairs have been highlighted which commonly give rise to intermolecular bonds that are unrealistically short, as well as a large problem with the method which leads to the whole system rotating during calculation. The best results are achieved using structures involving first row elements and a low proportion of the lighter halogens and chalcogens. In structures containing the heavier halogens and chalcogens, there is a tendency for the intermolecular forces to be overemphasised. However, there were no unreasonable structures predicted by MOPAC; the largest effect observed was the shortening of charge-transfer bonds.

The current version of MOPAC could be confidently used as a method to validate the structure of organic crystal structures; though users should be aware that the prediction for any structure that contains the heavier halogens or chalcogens, or a large proportion of the lighter halogens is likely to be less accurate than otherwise.

For inorganic structures, MOPAC generally preserves the symmetry well, despite having no specific functionality to do so. However, the structure prediction can fail catastrophically, giving results where all symmetry is lost. In some examples it appeared that this is a result of erroneous atom-pair parameters giving rise to very short bonds, which leads to a large distortion in the whole structure. In other examples this is not the case (*e.g.* the Ag<sub>2</sub>O structure shown in figure 5.18), and the cause has yet to be determined by Stewart. Several atom-pairs which give rise to short bonds and large loss of



symmetry have been highlighted.

At present, MOPAC could not be used confidently as a general method for validation of inorganic crystal structures. If MOPAC predicted a similar geometry, then this could be a good indication of a correct structure. If MOPAC predicted a different structure, then it would be difficult to know whether this was because the starting structure was erroneous, whether there is a low-energy phase change, or whether MOPAC is modeling the system poorly.

## Chapter 6

# High-throughput prediction of $^{13}\text{C}$ NMR chemical shifts by quantum-mechanical GIAO calculations

### 6.1 Introduction

This chapter describes a collaboration between myself, Peter Murray-Rust and Henry Rzepa in performing high-throughput prediction of  $^{13}\text{C}$  NMR chemical shifts using density-functional theory (DFT) methods. Rzepa had developed a modification to Bifulco’s protocol[221] for the prediction of  $^{13}\text{C}$  shifts, though it had not been applied to a wide range of systems. The work here describes my efforts to produce an automated workflow to perform high-throughput calculations on CamGrid using data obtained from an Open NMR shift database, NMRShiftDB[204, 205]. The results of the work are used to highlight the accuracy of both the protocol and the data, and the range of systems to which the protocol can reasonably be applied.

### 6.2 Nuclear Magnetic Resonance

Nuclear magnetic resonance (NMR) is a powerful technique for structure determination in solution since it can provide a wealth of data that can be related to chemical structure, conformation, and their relationship or inter-

action with the surroundings. The fundamental data provided by an NMR spectrum are chemical shifts ( $\delta$ ), which are used to obtain atom connectivity and spin-spin coupling constants ( $J$ ), from which stereochemistry can be deduced.

Despite all of the 2D and 3D spectroscopic methods available today, structural assignment of complex molecules (*e.g.* natural products) is still a significant challenge. To aid assignment, the prediction of NMR spectra by computational methods can be used. In particular,  $^{13}\text{C}$  chemical shifts are spread over a wide spectral range, are relatively insensitive to solvent effects, do not exhibit spin-spin couplings (owing to the low natural abundance) and are sensitive to steric and electronic influences in the structure. Matching  $^{13}\text{C}$  spectra can be an indication of identity between two compounds, and so accurate prediction of  $^{13}\text{C}$  spectra is a good test for compatibility.

## 6.3 Computational NMR

Computational NMR methods can be divided into two categories:

1. heuristic and machine learning algorithms, such as HOSE (Hierarchical Organisation of Spherical Environments) codes and neural networks (NN),
2. quantum mechanical methods.

The former rely on matching molecular fragment fingerprints to large databases of spectra for known structures. These are in use in many proprietary chemistry packages and have been shown to work to a high degree of accuracy for  $^{13}\text{C}$  chemical shifts. Recent results using NMRShiftDB (see below) as a survey set show the average error of two such programs to be around 2ppm[206]. As with semi-empirical QM methods, the accuracy of these methods depends on the data they are trained on. As a result, they are purely interpolative, and work less well for compounds with unusual or novel substructures that are not represented in the database.

The development of methods and codes for quantum mechanical calculations of NMR parameters, together with the ongoing growth of computing facilities has led to the study of a wide range of chemical problems[207, 208]. The gauge-including atomic orbital (GIAO)[209, 210] method is one of the most common approaches for calculating nuclear magnetic shielding tensors. It has been shown to provide results that are often more accurate than those calculated using other approaches of the same basis set size[211]. In many cases, in order to take into account correlation effects, post-Hartree-Fock calculations of organic molecules have been performed using DFT methods, which usually provide significant results at a relatively low computational cost. These methods have recently been widely applied to aid both structural assignment[212, 213] and reassignment[214] (notably hexacyclinol[215]).

Forsyth first demonstrated that an inexpensive computational method could be applied to a diverse group of small organic molecules with good accuracy[216]. He used an MM3 geometry and calculated the chemical shifts using GIAO with the B3LYP method and a specialized basis set. Using an empirical linear correlation, an average shift deviation of only 2.3ppm was achieved across the data set. Bifulco refined this approach and showed that the HF/6-31G(d) method gave good results for rigid non-polar compounds[217]. The method was subsequently extended to flexible molecules using a Boltzmann weighted average of the low-energy conformers[218]. Bifulco later compared a wide range of computational methods and found that using the mPW1PW91/6-31G(d,p) DFT method for calculating both the optimised geometry and chemical shifts gave the most accurate predictions[221].

A modified version of Bifulco’s protocol was used by Rychnovsky during his structural reassignment of hexacyclinol[215]. His analysis consisted of three steps:

1. the best geometry was identified using a Monte Carlo conformational search with the MMFF force field,
2. the geometry optimization was calculated using the HF/3-21G method,

3. the NMR chemical shifts were calculated with mPW1PW91/6-31G(d,p) and the self-consistent-reaction-field CPCM continuum solvation model for the NMR solvent.

Note that the method in the second step was altered not because it was more accurate, but because it was computationally less expensive. The average absolute deviation for the reassigned structure was shown to be 1.8ppm with a maximum deviation of 5.8ppm. Rzepa subsequently performed calculations on the reassigned structure using mPW1PW91/6-31G(d,p) for both geometry optimization and chemical shift calculation, and used a self-consistent reaction field correction for solvation. From this an average absolute deviation of 0.9ppm and a maximum of 2.8ppm were obtained[222]. The efficacy of this method was also shown in the work on reassigning the structure of the obtusallenes[214].

### 6.3.1 The Rzepa Protocol

After communication with Rzepa, it was decided that I would perform high-throughput calculations using his protocol (which shall be referred to as HSR0) to test its ability to reproduce known spectra for a range of small to medium-sized structures. HSR0 is implemented as a Gaussian03 workflow file containing three steps (as shown in figure 6.1):

1. geometry optimization using the ‘cheap’ RHF/STO-3G method,
2. more accurate optimization using the mPW1PW91/6-31G(d,p) method,
3.  $^{13}\text{C}$  chemical shift calculation using mPW1PW91/6-31G(d,p).

To perform the calculations, I needed access to a dataset of structures with:

- a connection table (preferably including 3D coordinates),
- the spectra,
- assignment of spectral peaks to atoms,

```

%chk=checkpoint-file.chk
#N RHF/STO-3G opt(loose)

Optimisation using UFF coordinates and STO-3G (more stable than PM3)
0 1
N -0.0897 1.5477 -0.1004
C 0.0827 0.2946 0.0756
C -1.2603 -0.3941 0.1731
... truncated ...

--Link1--
%chk=checkpoint-file.chk
#N rmpw1pw91/6-31g(d,p) geom=checkpoint opt guess=read

Optimisation using STO-3G coordinates and DFT
0 1

--Link1--
%chk=checkpoint-file.chk
# rmpw1pw91/6-31g(d,p) NMR scrf(cpcm,solvent=Acetone)

Calculating GIAO-shifts.
0 1

```

Figure 6.1: Gaussian03 workflow file implementing HSR0. The file consists of three linked calculations. Note that the Z-matrix has been truncated.

- the spectrum’s metadata (the solvent is required in the HSR0 workflow).

A promising solution is to use NMRShiftDB, which fulfills all these criteria. Not only that, but as the content is Open, we are also able to share the data. This allows us to undertake the experiment in an Open manner (see section 6.5), where all data and results are made available to the public.

### 6.3.2 NMRShiftDB

NMRShiftDB[204, 205] is an Open Source, Open Access, Open Content[223] database for organic structures and their NMR spectra. As of 2008-05-23 NMRShiftDB contains 21,498 structures with 25,091 spectra, the majority of which are for  $^{13}\text{C}$  nuclei. The content comes from manual scraping of published literature, donated third-party databases and individual submissions, all of which are peer-reviewed by humans before being entered into NMRShiftDB. The web interface provides functionality for spectrum and structure

prediction (based on HOSE codes) as well as searching spectra, structures and other properties.

As NMRShiftDB is an Open Content database, users are provided with methods to download the entire dataset in several forms[231], one of which is CMLSpect[29]. For instance, the (truncated) CMLSpect document for ketene is provided below:

```
<?xml version="1.0" encoding="UTF-8"?>
<molecule id="nmrshiftdb10008900" xmlns="http://www.xml-cml.org/schema">
  <atomArray>
    <atom id="a1" elementType="C" x2="0.0021" y2="-0.0041"
      x3="0.0021" y3="-0.0041" z3="0.0020" formalCharge="0" hydrogenCount="0"/>
    .....
  </atomArray>
  <bondArray>
    <bond id="b1" atomRefs2="a2 a1" order="D"/>
    .....
  </bondArray>
  <scalar dictRef="cdk:molecularProperty" title="Remark" dataType="xsd:string">
    NMRShiftDB 10008900 www.nmrshiftdb.org_dkfz_2003-04-28_06:56:35_0430
  </scalar>
  <spectrum id="nmrshiftdb10014421" moleculeRef="nmrshiftdb10008900" type="NMR"
    .....
    xmlns:cml="http://www.xml-cml.org/dict/cml">
    <conditionList>
      <scalar dataType="xsd:string" dictRef="cml:temp" units="siUnits:k">298</scalar>
      <scalar dataType="xsd:string" dictRef="cml:field" units="siUnits:hertz">
        Unreported
      </scalar>
    </conditionList>
    <metadataList>
      <metadata name="nmr:OBSERVENUCLEUS" content="13C"/>
    </metadataList>
    <substanceList>
      <substance dictRef="cml:field" role="subst:solvent" title="Chloroform-D1 (CDC13)"/>
    </substanceList>
    <peakList>
      <peak xValue="194.0" xUnits="units:ppm" peakShape="sharp" peakMultiplicity="S"
        id="p1" atomRefs="a1"/>
      .....
    </peakList>
  </spectrum>
</molecule>
```

The CMLSpect document contains a CML **molecule** element providing the connection table along with 3D coordinates (which were calculated using CORINA). Note however that explicit H atoms are not provided in NMRShiftDB, they are described using the **hydrogenCount** attribute on each atom bonded to H. Along with the **molecule** a **spectrum** element is provided which contains the spectral data and metadata, which may include:

- the observed nucleus,
- the temperature,
- the field, in Hertz, used for the NMR experiment,
- the solvent,
- a description of each peak in the spectrum, with links to the `atoms` in the `molecule` using the `atomRefs` attribute.

This is very useful, as all the data to define the structure and spectra is provided in one highly structured document. Once the calculations are performed, it is possible to add another `spectrum` element to each document to describe the results. This will provide simple access to each data item during analysis.

Note that without NMRShiftDB, automating this work would have been very difficult. There are no other public databases of experimental NMR spectra, so the source would have been limited to the published literature. Problems with extracting NMR spectra from article full-text are:

- It can be difficult to find a spectrum’s associated structure (*e.g.* numerals are often used throughout publications as brief references to long-named compounds).
- Once the structure is found, it is unlikely that a connection table will be provided. This must be obtained by either parsing an image of the structure, or by performing a name-to-structure conversion.
- Assignment of peaks to atoms is highly error-prone.

Another option would be to write a spider to scrape article supplemental data and hope that enough authors had provided the spectra in formats such as JCAMP[224]. However, there are no figures for the number of spectra published using this method.



## 6.4 Calculations

The overall workflow for performing the calculations is very similar to that used for MOPAC in section 5.4:

1. select suitable structures from NMRShiftDB,
2. create Gaussian03 workflow files for each selected structure,
3. create a Condor submit file for each job to be run,
4. upload the Gaussian03 workflow and Condor submit files to the Condor submit host,
5. submit the jobs to the Condor pool,
6. retrieve the Gaussian03 output files,

The discussion and implementation of steps 1, 2 and 3 are provided in the following subsections. The scripts and programs used for steps 4, 5 and 6 are identical to those used in the MOPAC workflow, and so will not be described here. As in the MOPAC workflow, the scripts and programs created for each step are executed manually. As in the MOPAC work, it would have been useful to have Gaussian03 output CML. However, it is a closed system and so its standard textual output must be converted to CML.

### 6.4.1 Structure selection

After downloading a copy of the CML version of NMRShiftDB, a Java class was written to read each CML document into `JUMBO` and select those with connection tables that:

- contain a subset of the atoms H, B, C, N, O, F, Si, P, S, Cl, Br and I,
- have a MW < 300,
- do not have a chain of more than two non-H atoms,
- have at least one CML `spectrum` for  $^{13}\text{C}$  NMR where:

- the solvent is provided,
- the number of carbon atoms in the spectrum is equal to the number in the structure.

This gave a set of rigid, small molecules, while still allowing for the presence of carbonyls and other functional groups. The first 400 structures to match these criteria were chosen to be used during this work.

### 6.4.2 Gaussian03 input

For each selected structure, the next step was to create a Gaussian03 workflow file in the form shown in figure 6.1. This is greatly eased by having the data in a consistent CML structure, from which data extraction is simple and robust using **JUMBO** (this is based on **XOM**, which contains an **XPath** library). Note that the only variables in the workflow are the path to the checkpoint file, the Z-matrix in the first step and the solvent in the third. To create the files, a simple Java class was written to iterate through each structure, performing the following steps:

1. read the CML into **JUMBO**,
2. convert the implicit H atoms into explicit **atom** elements,
3. create 3D coordinates for each H atom (**JUMBO** contains methods for this),
4. convert the **atomArray** into a Gaussian03 Z-matrix,
5. extract the solvent from the **spectrum**.
6. enter the Z-matrix and solvent into the workflow template shown in figure 6.2. The checkpoint file is assigned to have the same name as the input CML file (its MIME is provided in the workflow template),
7. output workflow out with the same name as the input CML file, though with the Gaussian03 workflow MIME, **gjf**.

```

%chk=<checkpoint-path>.chk
#N RHF/STO-3G opt(loose)

Optimisation using UFF coordinates and STO-3G (more stable than PM3)
0 1
<z-matrix>

--Link1--
%chk=<checkpoint-path>.chk
#N rmpw1pw91/6-31g(d,p) geom=checkpoint opt guess=read

Optimisation using STO-3G coordinates and DFT
0 1

--Link1--
%chk=<checkpoint-path>.chk
#P rmpw1pw91/6-31g(d,p) geom=checkpoint guess=read
#P NMR scrf(cpcm,solvent=<solvent>)

Calculating GIAO-shifts.
0 1

```

Figure 6.2: A template for Gaussian03 input files implementing HSR0. The three variables <checkpoint-path>, <z-matrix> and <solvent> must be replaced.

### 6.4.3 Condor input

The Condor submit files for the Gaussian03 jobs (as in figure 6.3) are similar to those used for the MOPAC work. To create them, a minor alteration was made to the program used in the MOPAC work. As Gaussian03 accepts an argument signalling where the the output file must be written to, there is now a line containing the `output` parameter. Note also that, as Gaussian03 is installed on a number of machines in the CamGrid, that I do not have to provide the executable with each job. Instead, I can specify that the jobs run only on those machines with Gaussian03 by using:

```
HAS_GAUSSIAN == TRUE
```

in the `requirements` section.

A number of tests were run prior to the batch jobs to ensure that they would execute successfully. During this testing, it was noticed that all jobs submitted to three of the compute nodes were failing immediately. After

```

universe=vanilla
getenv=True
requirements = Arch == "X86_64" && OpSys == "LINUX" && HAS_GAUSSIAN == TRUE
executable = /home/ned24/gaussian/everything/1/nmrshiftdb10003239-1.sh
input = /home/ned24/gaussian/everything/1/nmrshiftdb10003239-1.gjf
output = nmrshiftdb10003239-1.out
error = nmrshiftdb10003239-1.err
log = nmrshiftdb10003239-1.log

should_transfer_files=YES
when_to_transfer_output=ON_EXIT_OR_EVICT

Queue

```

Figure 6.3: An example Condor submit file for Gaussian03 jobs.

investigation, this was shown not just to apply to Gaussian03 jobs. To make sure that no further jobs were submitted to these nodes, the following was appended to the `requirements` line in the submit file:

```

&& Machine != "gridlock20--ch.grid.private.cam.ac.uk"
&& Machine != "gridlock26--ch.grid.private.cam.ac.uk"
&& Machine != "gridlock27--ch.grid.private.cam.ac.uk"

```

Interestingly, it was noted that for these commands to be applied by the system, they would have to be placed before the `HAS_GAUSSIAN` parameter.

## 6.4.4 TMS

Tetramethylsilane (TMS) is the usual standard to which chemical shifts are related. These chemical shifts,  $\delta(^{13}\text{C}_i)$ , are defined by:

$$\delta(^{13}\text{C}_i) = \sigma(^{13}\text{C})_{\text{TMS}} - \sigma(^{13}\text{C}_i)$$

where  $\sigma$  is the nuclear isotropic shielding tensor. To have accurate calculated chemical shifts, the geometry and the isotropic shielding constant of carbon atoms in TMS were computed at the same level of calculation as used for the structures in the dataset. As a workflow is being used where the solvent must be specified, the calculation was performed on TMS using all solvents in the dataset. All calculated chemical shifts discussed in the analysis will be with reference to the average chemical shift of the carbons in TMS in the same solvent.

## 6.5 Open Computational NMR

Before undertaking this investigation, it was decided that it would be performed as an Open experiment, where the method and results would be made public. The initial intention was to perform the experiment as Open Notebook Science[225] (ONS), where all methods and results are published in as close to real-time as possible on the Web. During the experiment the results were published as they were discovered, though as the methodology was not made clear from the outset, the experiment could not be classed as ONS. It was instead given the title of ‘Open Computational NMR’.

One of the core benefits of publishing in such a way is that it allows *crowdsourcing*, described as the solution of problems through a distributed network of people. That is, the more people looking at the data, the more chance there is of finding errors or relationships; to borrow a phrase from Open Source software development:

Given enough eyeballs, all bugs are shallow.[227]

It was decided that the best method of publishing the experiment would be to provide a webpage with the full, continuous experimental commentary[226], and to notify others of each individual finding as it happened via a blog (written by Peter Murray-Rust[228]\*). As blogs allow user comments, this would also serve as a good place for community discussion and feedback.

To publish the results I could just create a webpage that provides hyperlinks to all output data. This would be most unhelpful however, as the average reader would not have the time or inclination to analyse it themselves. Likewise, if only the numerical results of analysis were provided, *e.g.* the average shift deviation, users would be denied the opportunity to view each data point and discuss the origins of the deviations. It is therefore important to provide interactive visualizations that allow linking of graphical points back to the original data.

---

\*All posts regarding this experiment are provided in the ‘NMR’ category of this blog at <http://wmm.ch.cam.ac.uk/blogs/murrayrust/?cat=22>

To allow this, a program was written that could read a series of CMLSpect files and automatically produce a webpage like the one in figure 6.4. The graphs on these pages are created as SVG (Scalable Vector Graphics) using a Java tool developed by Townsend[165]. JavaScript is then embedded into this SVG to allow points to become interactive. By embedding a Jmol applet into the same webpage, it is possible to allow the optimised geometry of the structure related to each point to be displayed upon clicking<sup>†</sup>. Note that for each point clicked on the above page, the associated carbon atom is also highlighted in the Jmol applet by a yellow halo. Links to other visualizations, such as plots for individual structures are also provided, which then give links back to the original data on the NMRShiftDB website. For each piece of analysis performed, an interactive webpage was created showing the results, and a link to it would be provided on the experiment commentary webpage.

## 6.6 Analysis

### 6.6.1 Preparing the output

Of the 400 structures selected for calculation, 105 of the geometries were not successfully optimized before the calculation cycle limit of 100 was reached. Rather than increase the cycle limit and recalculate, it was decided to continue with the 295 structures that were successfully calculated.

In order to ease the analysis, a Java program was written to extract the chemical shifts and optimized Cartesian coordinates from each successful Gaussian03 output file (as in figure 6.5), convert them into a CML **spectrum** and then merge this with the original NMRShiftDB CML file. This means that both observed and calculated shifts for each structure are in a common format within the same document, and also allows the CML file to be read into Jmol to display the optimized geometries side-by-side with the interactive plots.

---

<sup>†</sup>Note that these graphs are currently only interactive when viewed with the Mozilla Firefox browser.

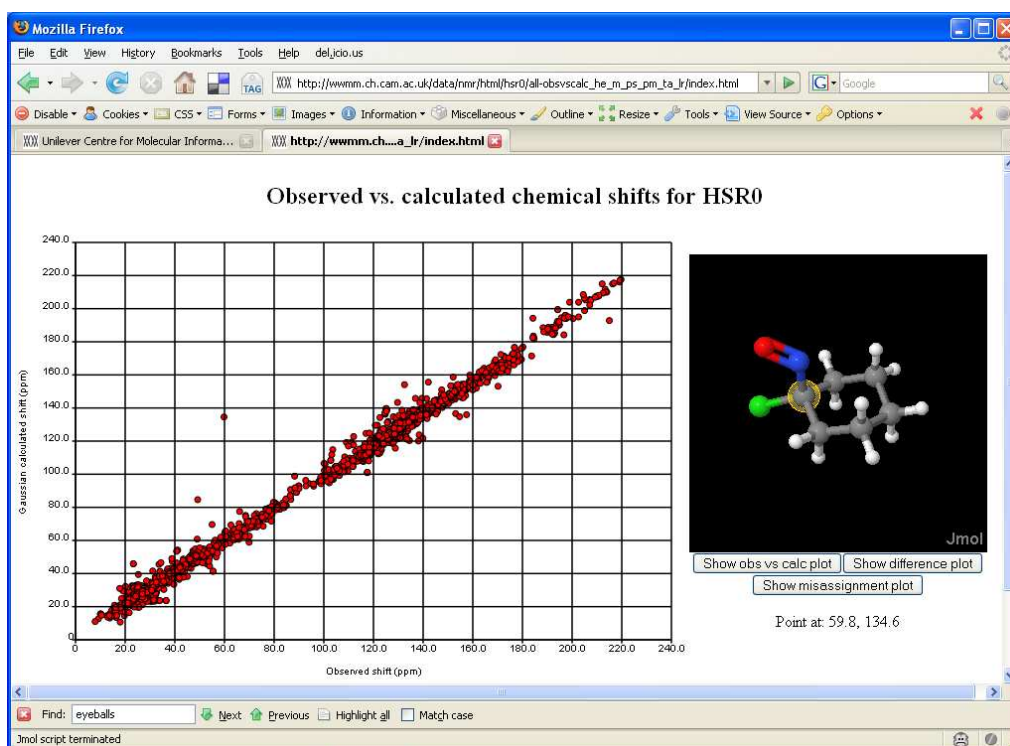


Figure 6.4: Screen capture of the webpage at [http://wwmm.ch.cam.ac.uk/data/nmr/html/hsr0/all-obsvsalc\\_he\\_m\\_ps\\_pm\\_ta\\_lr/index.html](http://wwmm.ch.cam.ac.uk/data/nmr/html/hsr0/all-obsvsalc_he_m_ps_pm_ta_lr/index.html). Note that this plot is equivalent to the one shown in figure 6.6.

```

Calculating GIAO nuclear magnetic shielding tensors.
SCF GIAO Magnetic shielding tensor (ppm):
 1 N Isotropic = -39.7361 Anisotropy = 217.7293
   XX= -59.3942 YX= 24.3029 ZX= -4.0078
   XY= 44.0071 YY= -162.8265 ZY= 26.0997
   XZ= 0.8510 YZ= 24.5147 ZZ= 103.0123
   Eigenvalues: -175.2799 -49.3452 105.4168
 2 C Isotropic = 62.1414 Anisotropy = 88.4926
   XX= 65.2163 YX= 61.2839 ZX= -6.7794
   XY= 8.8132 YY= 1.1062 ZY= 10.5170
   XZ= -4.6164 YZ= 9.9715 ZZ= 120.1018
   Eigenvalues: -15.3413 80.6292 121.1365
 3 C Isotropic = 134.0758 Anisotropy = 39.0721
   XX= 122.3085 YX= 17.7797 ZX= 0.7534
   XY= 10.9514 YY= 153.7939 ZY= -5.6388
   XZ= -4.3483 YZ= -3.8754 ZZ= 126.1248
   Eigenvalues: 116.7390 125.3645 160.1238
 4 N Isotropic = 215.1393 Anisotropy = 84.4722
   XX= 264.2717 YX= 21.4197 ZX= 1.2077
   XY= 18.3645 YY= 216.3611 ZY= -5.7668
   XZ= 9.0644 YZ= -22.8941 ZZ= 164.7850
   Eigenvalues: 160.0664 213.8973 271.4541
 5 C Isotropic = 105.0617 Anisotropy = 49.0603
   XX= 102.3425 YX= -19.7825 ZX= -0.5348
   XY= -17.7754 YY= 127.0115 ZY= -8.4856
   XZ= 1.6121 YZ= -3.8549 ZZ= 85.8312
   Eigenvalues: 84.4413 92.9753 137.7686

```

Figure 6.5: Section of a Gaussian03 output file showing the calculated isotropic magnetic tensors.

## 6.6.2 Initial results

The plot showing all 2341 calculated versus observed  $^{13}\text{C}$  shifts is shown in figure 6.6. The shifts for nuclei bonded to chlorine and bromine are shown to be routinely overestimated during calculation. This is a well-known effect due to spin-orbit (SO) coupling and is exhibited increasingly by ‘heavy elements’[229]. While calculation of the SO effect is possible using third-order perturbation theory[230], additive corrections have been shown by Rzepa to give accurate results[214]. Rzepa supplied these offsets to apply to carbon atoms bonded to S, Cl and Br, which are -2, -3 and -12ppm respectively. The plot for the same set of shifts after these offsets have been applied is shown in figure 6.7. This shows a noticeable improvement in the 100-150ppm range and removes a gross outlier at around (50,80) which corresponded to a carbon bonded to two bromine atoms. The average absolute deviation for calculated versus observed shifts is improved from 3.25ppm to 3.15ppm on inclusion of these offsets.



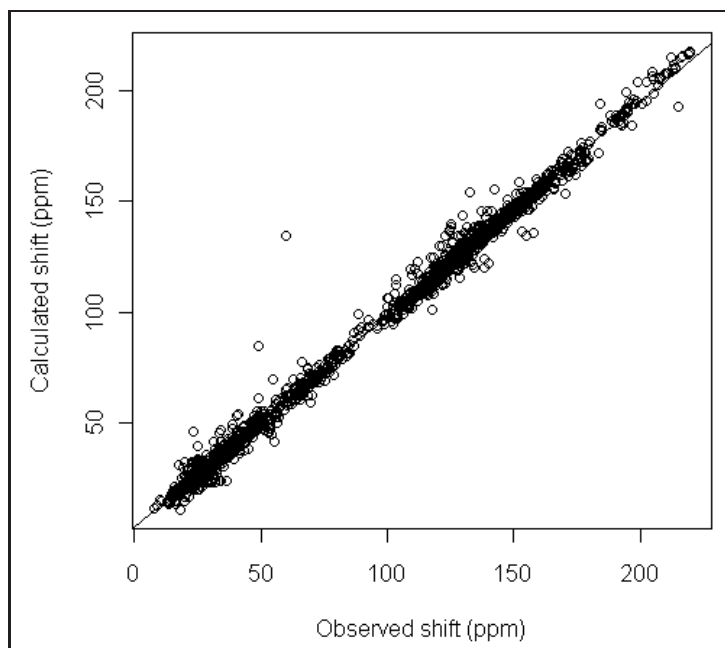


Figure 6.6: Plot showing all calculated versus observed  $^{13}\text{C}$  shifts for the 295 successfully completed calculations using HSR0.

### 6.6.3 Sources of error

All datasets contain errors, and so it is helpful to know their causes in order to aid discovery and remedy. There are many possible sources of error in collecting NMR spectra and assigning shifts, some of which are discussed below.

Experimental or reporting errors are independent of the prediction of the effect. These include:

- mis-reported solvent (the shifts are solvent dependent and the calculation tries to simulate this)
- variable calibration of the NMR instrument (*e.g.* giving rise to origin shifts)
- impure compound — the sample may contain a substance which gives rise to appreciable peaks not belonging to the title compound

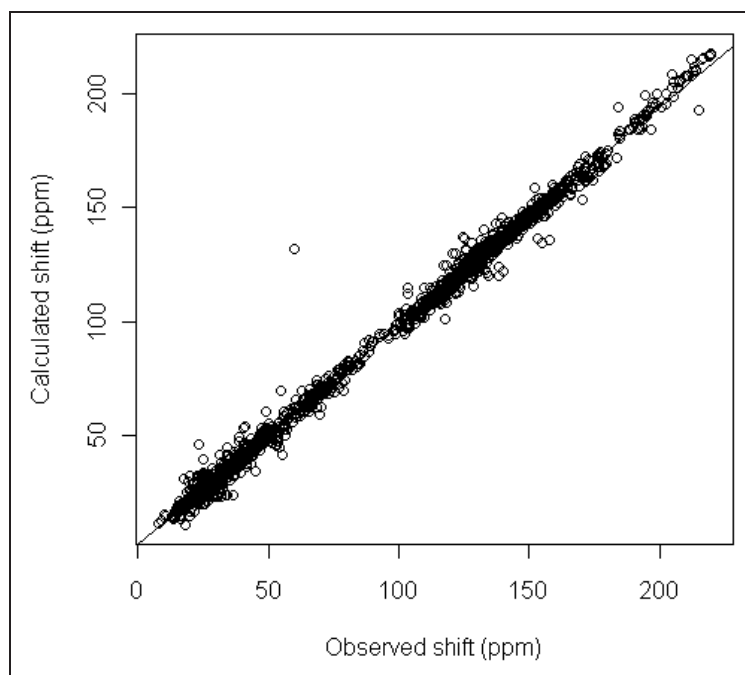


Figure 6.7: Plot showing all calculated versus observed  $^{13}\text{C}$  shifts for the 295 successfully completed calculations using HSR0 after spin-orbit offsets have been applied. The outlier at (60,135) is explained in section 6.6.4.

- wrong compound assigned to spectrum (*i.e.* error in bookkeeping or drawing error)
- machine parameters (*e.g.* field strength) varied or reported incorrectly
- transcription errors in spectrum or peaks
- misassignment of peaks to inappropriate atoms
- broad peaks with considerable variance leading to misreporting of mean (this is unlikely in  $^{13}\text{C}$  owing to the sharpness of the peaks)
- errors in applying theory of NMR or its interpretation
- noise (including random noise and mains spikes)
- human editing of spectra

A prediction error is independent of the reported value for the shift. Some are theoretical, some are computer ‘bugs’. These include:

- mis-calculation of offset (*e.g.* from isotropic tensor to observed shift)
- mis-assignment of calculated peaks to atoms
- corruption of connection tables (particularly in the adding of H atoms)
- mismapping of atoms between input and output of calculation. As there is no way to add IDs to atoms in the program used, I must rely on the atoms being output in the order they are provided in the input
- incorrect generation of program input
- program bugs in reading input and main calculation (*e.g.* it was reported[165] that for GAMESS input, if a line overflowed 80 characters then the atom on the following line was reported but not included in the calculation)
- incorrect transformation of output to CMLSpect

- theoretical model has limitations
- oversimplified chemical model. Some common problems are:
  - only one conformer is calculated
  - symmetry is not well treated
  - tautomerism is ignored
  - isomerism (*e.g.* ring-chain is ignored)

There are also potential bugs on the computational side, such as:

- inconsistent results from different machine architectures
- errors in processing and outputting the results

As the starting dataset is the list of assigned peaks for each structure, and not the spectrum obtained from the NMR machine, means that some errors, even if discovered, cannot be assigned to a cause with full confidence. For instance, if I have data from an in-house database where it appears that the structure for a given set of peaks may be wrong, how do I know if this is the case, or whether a shift has been incorrectly supplied? Similarly, knowing whether the correct offset has been applied to an observed spectrum can only be asserted by having access to the original spectrum. To enable detailed analyses of computational NMR methods, access is required to databases of not just assigned peak lists, but also the original spectra. However, at the present time, this is not feasible.

#### 6.6.4 Cleaning the dataset

The ideal dataset for this work would not contain any errors in the creation or assignment of the NMR shifts, though in reality this is unattainable. Discovering and fixing errors in datasets is a key part of eScience, and has already been discussed with regards to NMRShiftDB by Robien[232] and Blinov *et al.*[206] using HOSE/NN methods. In performing this work, I had the opportunity to use the output of the calculations to discover and

report errors in NMRShiftDB. As the results were made public with interactive visualizations, the interested community (which included maintainers of NMRShiftDB) could help to do this.

To aid the location of ‘problem’ structures, an interactive plot was created to show the RMS deviation versus the mean deviation for the shifts of each structure (see figure 6.8). This allowed users to click through the points corresponding to high RMS deviation or mean deviation and then inspect plots for those structures alone. In addition to this, ‘misassignment’ plots were created for each structure where the difference in observed and calculated shifts was plotted against the average of the two shifts. Plotting the data in this way can help identify those shifts that may have been misassigned by finding two points with the same average shift and equal but opposite shift difference (see figure 6.9).

Through community discussion via Peter Murray-Rust’s blog, and the use of the above interactive plots, the data for 7 structures were initially found to be erroneous:

- The observed shift for a carbon in nmrshiftdb10006060 (the NMR-ShiftDB ID for the structure) was shown by Rzepa to have been incorrectly copied from the published literature (this point is the largest outlier shown at around (60,135) in figure 6.7).
- The structures with the highest and lowest mean deviations (nmrshiftdb10009121 and nmrshiftdb10006328) corresponded to tautomers taken from the same sample. Christoph Steinbeck (head of the NMR-ShiftDB project) highlighted that the shifts for these had been taken from the same spectrum, but some of them had been misassigned to the wrong tautomer.
- Two structures (nmrshiftdb2275 and nmrshiftdb2562) were judged to have an incorrect structure for the shifts that had been supplied.

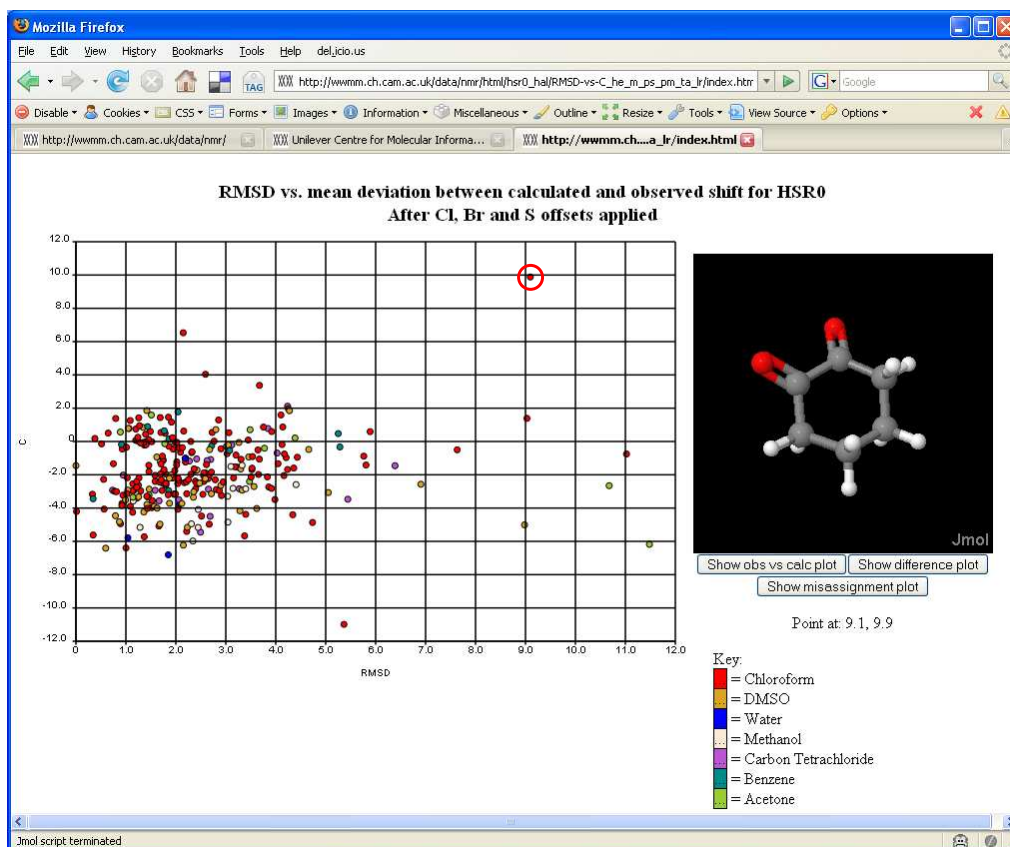


Figure 6.8: Screen capture of the webpage at [http://wwmm.ch.cam.ac.uk/data/nmr/html/hsr0\\_hal/RMSD-vs-C\\_he\\_m\\_ps\\_pm\\_ta\\_lr/index.html](http://wwmm.ch.cam.ac.uk/data/nmr/html/hsr0_hal/RMSD-vs-C_he_m_ps_pm_ta_lr/index.html). The plot shows the RMS deviation vs. mean deviation for all structures calculated using HSR0 with spin-orbit offsets applied. The circled point corresponds to the structure shown in the Jmol applet, the data for which were subsequently shown to be incorrect. Note that the ‘misassignment’ plot for any structure shown in the Jmol applet can be viewed by simply clicking a button beneath the applet.

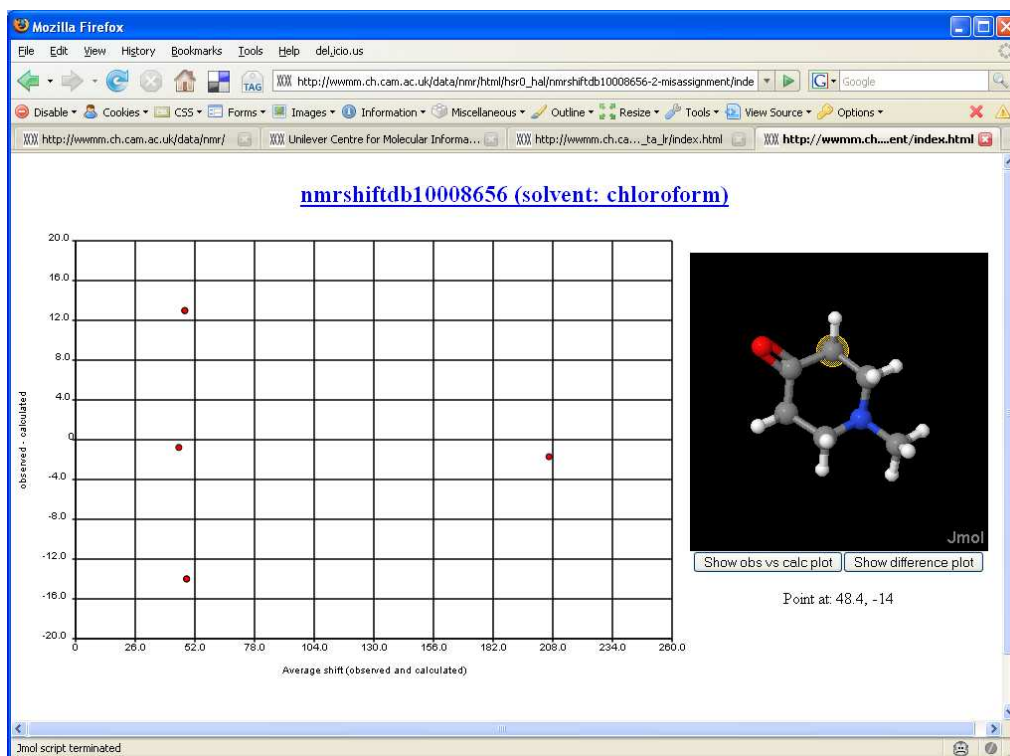


Figure 6.9: Screen capture of the webpage at [http://wwmm.ch.cam.ac.uk/data/nmr/html/hsr0\\_hal/nmrshiftdb10008656-2-misassignment/index.html](http://wwmm.ch.cam.ac.uk/data/nmr/html/hsr0_hal/nmrshiftdb10008656-2-misassignment/index.html). This 'misassignment' plot shows the difference in observed and calculated shift plotted against the average of the two. Those two shifts with equal but opposite differences between observed and calculated shift and equal average shift were shown to be misassigned.

- Two structures (nmrshiftdb10008656 and nmrshiftdb10006416) were judged to have shifts that had been misassigned.

These 7 structures were subsequently removed from the dataset.

In order to provide simple access to those structures that had possible misassignments, a program was written to find structures with two peaks for which the average of observed and calculated shifts were within 2ppm, and the difference between the observed and calculated shift was greater than 2ppm. The program found 42 structures with potential misassignments, all of which were then removed from the dataset before further analysis took place (which left 246 structures with 1943 peaks). After a list of these structures was made available via the experiment commentary webpage, a member of the NMRShiftDB team inspected the data for the 42 structures and found 11 of them to be incorrect[233]. This included 10 structures with misassigned peaks and 1 incorrect structure. Thus, 18 out of the initial 295 structures were shown to contain errors, which corresponds to an error rate of 6.1%. Note that all data found to be incorrect during this work has now been corrected in NMRShiftDB.

### 6.6.5 Conformational issues

As I am only calculating the chemical shifts for the conformer that is calculated to be most stable for each molecule, it is possible for chemically equivalent atoms to have two different calculated shifts (such as in figure 6.10). This effect can be removed by comparing the Morgan number[234] for each carbon in a structure, and averaging the shifts of those that are equivalent. The Morgan numbers can be calculated from a CML document using the `Morgan` class in `JUMBO`. An example of the shift averaging for chemically equivalent atoms can be seen by comparing figures 6.10 and 6.11.

This, however, only fixes conformational problems for chemically equivalent atoms. If, for instance, there was a substituent in the *meta*-position to the vinyl group in the structure in the previous plots, then the two atoms



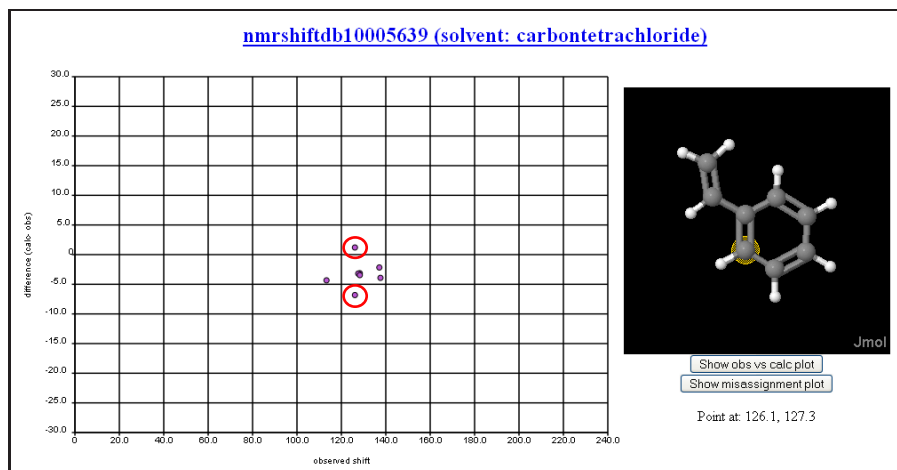


Figure 6.10: Plot showing the difference in calculated and observed shifts against the observed shift for styrene. The two circled points are those for the two carbons  $\alpha$  to the vinyl group. These are chemically equivalent, though the geometry optimization has meant they are inequivalent when the isotropic magnetic tensors are calculated.

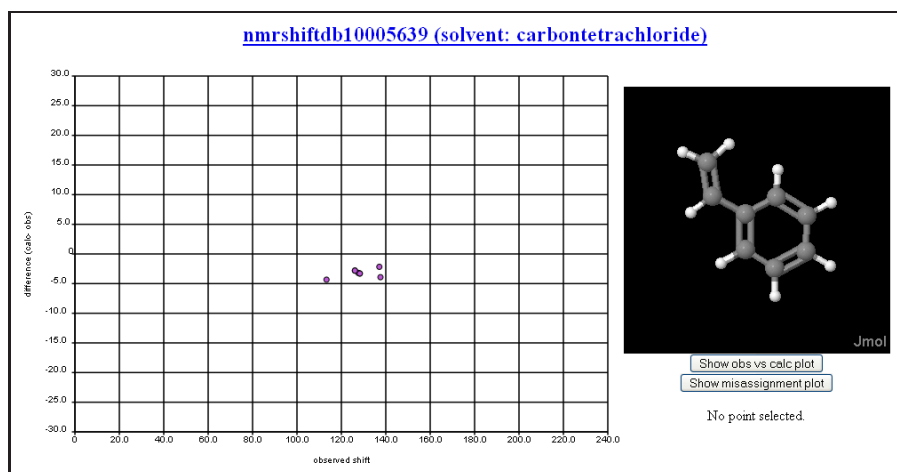


Figure 6.11: Figure showing the same plot as in figure 6.10, though this time with averaging of the shifts for Morgan equivalent atoms.

in the *ortho*-position to the vinyl group would no longer be equivalent. It is likely that each of them, in the most stable calculated conformation, would be experiencing the effects of the vinyl group more or less than they would on average if free rotation were allowed. Thus, the chemical shifts for both will be incorrect, and the only way to counter this effect is to calculate the set of most highly occupied conformers and use a weighting to average the calculated shifts for each. This has been discussed in work by Bifulco[218], who showed that conformational analysis and Boltzmann averaging can be used for accurate shift predictions of flexible molecules.

Due to time constraints, I was not able to undertake such analyses during this work and tried to select only rigid molecules to avoid this problem. In order to minimise these effects further, it was decided that I would remove all structure with 7-membered rings or larger, and those structures that were possibly tautomeric. After this filtering, 207 structures were remaining with 1544 peaks.

### 6.6.6 HSR1

Rzepa provided a modified version of HSR0 (hereafter called HSR1). This incorporated an improved 6-31G(d) basis set in the NMR shift calculation to improve the values calculated for vinyl and carbonyl carbon atoms. This basis set included an additional diffuse ‘3p’ function for O and C, estimated from the corresponding values for S and Si. The HSR1 template is shown in figure 6.12. For the 207 remaining structures and TMS, the calculations were rerun using this new protocol. As the first two steps of both protocols are identical, the checkpoint files from the previous calculation could be used so that only the shift calculation was rerun in each case. Figure 6.13 shows the calculated versus observed shifts after both spin-orbit offsets and Morgan averaging have been applied to calculations from HSR1.

Table 6.1 shows the improvement of the average absolute shift deviation on application of the spin-orbit coupling offsets and averaging of the shifts for topologically equivalent atoms for both HSR0 and HSR1 protocols. The

```

%chk=<checkpoint-path>.chk
#N RHF/STO-3G opt(loose)

Optimisation using  UFF coordinates and  STO-3G (more stable than  PM3)
0 1
  <z-matrix>

--Link1--
%chk=<checkpoint-path>.chk
#N rmpw1pw91/6-31g(d,p) geom=checkpoint opt guess=read

Optimisation using STO-3G coordinates and DFT
0 1

--Link1--
%chk=<checkpoint-path>.chk
# rmpw1pw91/6-31g(d,p) NMR scrf(cpcm,solvent=<solvent>) ExtraBasis

Calculating  GIAO-shifts.
0 1
C      0
SP      1      1.00
          0.05      1.00000000      1.00000000
****
O      0
SP      1      1.00
          0.070000      1.0000000      1.0000000
****

```

Figure 6.12: A template for Gaussian03 input files implementing HSR1. Note that the second extra basis set in the final step should not be included if the structure does not contain any oxygen atoms.

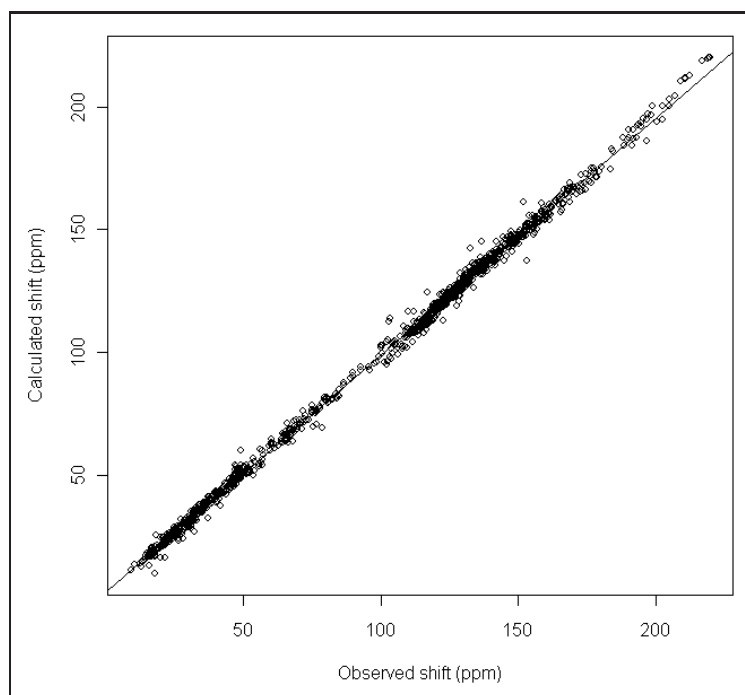


Figure 6.13: Calculated vs. observed chemical shifts for the HSR1 protocol after spin-orbit offsets and averaging of the shifts for topologically equivalent atoms have been applied.

	<b>intercept</b>	<b>slope</b>	<b><i>p</i></b>	<b>average abs. deviation (ppm)</b>
HSR0	2.237	0.960	0.998	2.94
+ spin-orbit offset	2.093	0.960	0.999	2.84
+ equiv. atom averaging	2.093	0.960	0.999	2.80
HSR1	2.435	0.965	0.998	2.68
+ spin-orbit offset	2.291	0.965	0.999	2.54
+ equiv. atom averaging	2.291	0.965	0.999	2.50

Table 6.1: The linear fitting coefficients (where  $p$  is the Pearson correlation coefficient) of calculated versus observed shifts for the remaining 207 structures using HSR0 and HSR1. Also provided is the average absolute deviation of each shift. For each protocol, the statistics are also provided for subsequent application of the spin-orbit coupling offsets and averaging the shifts for topologically equivalent atoms.

intercepts in each case improve on application of the spin-orbit offsets. It has been shown elsewhere that the slope of the linear correlation between observed and calculated shifts can deviate more or less than 1 depending on the computational method and choice of basis set[216, 219, 220]. As expected, averaging the shifts for topologically equivalent atoms does not affect the linear fitting coefficients. For the final average absolute deviations, there is an 11% improvement using HSR1 over HSR0.

It is useful to have shown that the steps taken have led to improvements between and within the HSR0 and HSR1 protocols, though as the dataset has been shown to contain a significant rate of errors, it is difficult to be confident of their accuracy when comparing to other methods. However, while cleaning the dataset, it was noticed that many of the errors came from structures donated by the same source. The source metadata is not provided in each CML file, though the field strength is, and since the spectra from this provider were created at an unusual field (23Hz) I could filter the structures using that. After removing all spectra taken at that field strength, only 36 structures remained (containing 317 peaks). The calculated versus observed shift plot for these 36 structures is shown in figure 6.14. The linear fitting coefficients and average absolute deviation using HSR0 and HSR1 with

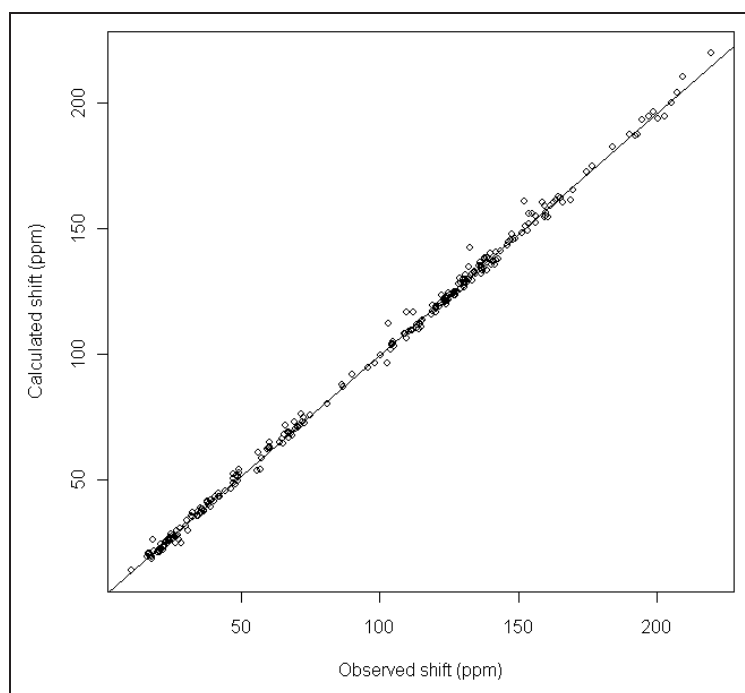


Figure 6.14: Calculated vs. observed shifts for HSR1 with spin-orbit offsets and averaging shifts for topologically equivalent atoms for those structures with spectra determined at a field of over 25Hz.

spin-orbit offsets and averaging of shifts for topologically equivalent atoms are shown in table 6.2. By analysing only these 36 structures, there are improvements of 17% and 10% in the average absolute shift deviations using HSR0 and HSR1 respectively.

### Accuracy of the spin-orbit offsets

For the remaining 36 structures, there were 19 shifts for which spin-orbit offsets had been applied. For carbon atoms bonded to S, Cl and Br, there were 11, 4 and 1 shifts respectively. The average deviation and average absolute deviation for these shifts were 0.23 and 1.66. For carbon atoms bonded to sulphur alone, these were 0.10 and 1.70, and to chlorine they were 0.72 and 1.77. These figures show that the shifts for which spin-orbit offsets have been applied are more accurate than the overall dataset. The results for the shifts of carbon bonded to chlorine suggest that an offset of -3.5ppm may

	<b>intercept</b>	<b>slope</b>	<b><i>p</i></b>	<b>average abs. deviation (ppm)</b>
HSR0	2.477	0.963	0.999	2.36
HSR1	2.708	0.967	0.999	2.24

Table 6.2: Linear fitting coefficients and average absolute shift deviation using only those spectra which were taken at fields of over 25Hz.

provide more accurate results than the -3ppm offset used, though as there are only 4 data points this would need to be investigated more thoroughly to be justified.

### 6.6.7 Conclusions

The NMRShiftDB is a helpful resource for the purpose of evaluating chemical shift prediction accuracy, as evidenced by this work and the previous work of Blinov *et al.*[206] and Robien[232]. As has been discussed in this and the previous works, there are outliers in the dataset requiring review and correction. Blinov has shown that the literature itself contains around 8% errors in the form of mis-assignments, transcription errors and incorrect structures, and 6.1% of the structures used in this work were shown to be erroneous. Despite this, this error rate is far higher than desired, and it would be useful to introduce robotic refereeing of submissions to NMRShiftDB, either by QM or HOSE/NN methods to help reduce the error rate.

Performing this work in an Open manner was very beneficial in analysing the quality of the dataset. As public web-based tools were provided to allow inspection of the data, most of the errors discovered were described by people not directly involved in the work (thus demonstrating the power of crowd-sourcing!). As all discussion was held in public on the Web, this allowed the maintainers of NMRShiftDB to immediately find and fix any uncovered errors.

The data presented here shows that HSR1 with spin-orbit offsets and Morgan averaging has an average deviation in predicted  $^{13}\text{C}$  chemical shifts of

less than 2.25ppm for small, rigid structures. This is comparable to the results obtained for the entire NMRShiftDB dataset using HOSE (2.22ppm) and NN (1.59ppm) methods.

Nominally, the far greater cost of the QM approach compared to HOSE/NN was not worth while, given the comparable errors. However, the QM approach relies on no library of chemical shifts, as HOSE/NN do. This makes the QM approach more extrapolative, whilst the HOSE/NN methods are interpolative. The former would always be preferable for compounds with unusual or novel structures, or structures where a conventional group has been strained or distorted from its normal structure.



# Chapter 7

## Conclusions

As stated in section 1.7, the goals of this work were twofold; to aid the creation of a Chemical Semantic Web and to show how Open, semantic data is beneficial to e-Science. The work performed for the former can be viewed as two parts:

1. An assessment of the ability to locate chemical structures on the Web using current chemical identifiers as query strings.
2. The creation of CrystalEye, a web-based, Open collection of semantic crystallographic data, which includes a number of related services.

The work performed for the latter demonstrated two examples of using Open data to perform high-throughput e-Science experiments.

The InChI and SMILES identifiers now provide widespread, robust methods for exact and sub-structure searches for organic species. These solutions are sufficient for searching over a large proportion of Web-based chemical databases (*e.g.* PubChem, ChemSpider). However, while these identifiers can also represent organometallic structures, they are unable to uniquely identify them in the same way as with organics. For instance, the default setting in InChI is to disconnect all bonds to metal atoms, and not to treat them at all in the canonicalization algorithm[238]. This arises from the fact that there are no standards between chemists for depicting the bonding in

organometallic molecules, and hence there is no way to implement a canonicalization algorithm. It is possible to generate an InChI for an organometallic structure which contains an extra layer (the 'reconnected layer') which uses the bonds in the input structure[238]. However, this will lead to problems in having more than one identifier for the same structure if two people were to use different metal-ligand bonding schemes in their input structures.

There are large number of organometallic structures in current crystallographic literature, and hence a unique identifier for them is important for searching of databases such as CrystalEye. There are no clear solutions to this problem at present, though I think the pragmatic approach to the problem is currently to use the default settings of InChI, and alert any searchers that queries for organometallics may return a small number of false positives.

Crystallography is unusual in the chemical community, in that the experimental data is provided Openly alongside published articles as supplemental information in a community standard with well-defined syntax and semantics. As shown, there is an abundance of crystallographic data available on publishers websites that is technically available for reuse by anyone with an Internet connection, though in practice, reuse is more difficult. This is because the data is not actively promoted by the publishers, but is still largely seen as a method for validation of the article during the peer-review process. The implementation in CrystalEye shows that it is possible to aggregate this data with high recall and precision by spidering HTML pages, though the process is fragile and highly susceptible to changes in website structure. All publishers now use RSS/Atom feeds to notify readers of new articles, and as described, simple additions to these feeds would also allow notification of data provided alongside these articles. As RSS/Atom are XML-based and are widely adopted throughout the Web community, robust software to aid automated creation and parsing of these is Openly available. Hence, this method would be an excellent choice for persistent data distribution by publishers.

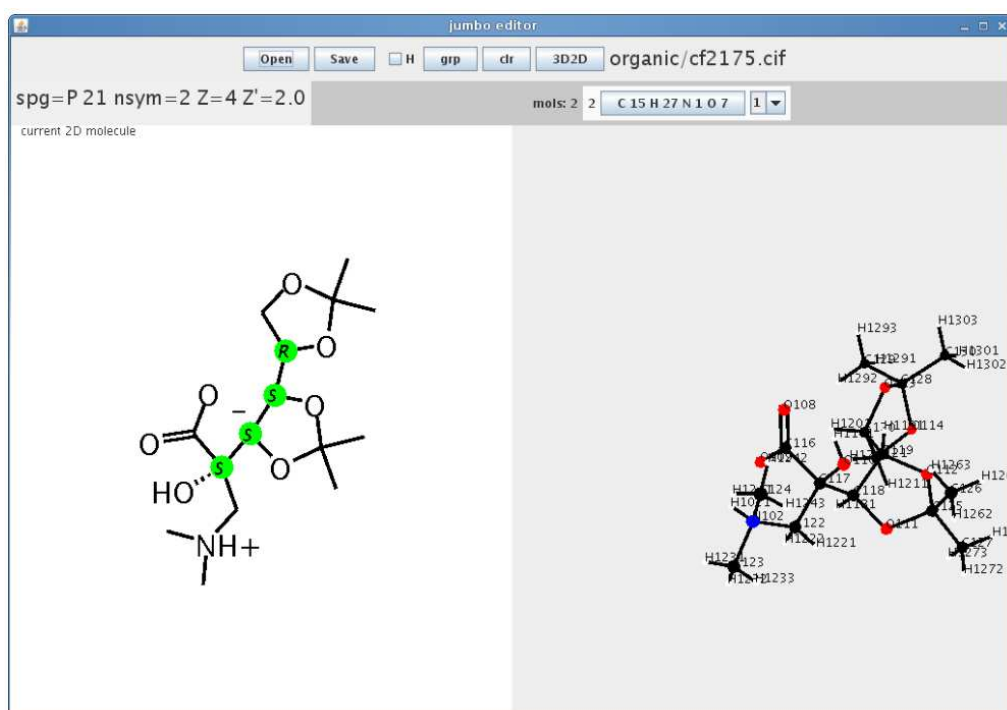
However, only around 20% of all crystal structures are published, and institutions are now beginning to capture this data. An example of this is the University of Southampton's eCrystals site[52], where data for crystal structures generated by the Southampton Chemical Crystallography Group and the EPSRC UK National Crystallography Service are made available. We are also now investigating the capture of unpublished crystal structures from the Department of Chemistry at the University of Cambridge using software based on CrystalEye[235]. In order to gain the most value from the new generation of stores being created, the eCrystals project[236] is due to provide infrastructure to enable a pan-institutional federation of crystallographic repositories (of which CrystalEye will be a member). The increasing rate of published crystal structures (as highlighted in figure 4.12) coupled with the fact that data for unpublished CIFs is now starting to be captured and shared will lead to a dramatic rise in the amount of crystallographic data Openly available. Indeed, it is not unreasonable to think that CrystalEye, which has data for around 140,000 crystal structures from the past 15 years, could double in size over the next 2-3 years.

There are costs to using unpublished data, and a primary concern is the quality of data that has not been checked by trained scientists or been through the publication process. The amount of data being currently produced is far too great to be checked completely manually, and thus computational validation methods such as those described in chapters 5 and 6 are required. Again, the crystallographic community is a forerunner in this field, in that CheckCIF, the validation tool used during the peer-review process for many journals, is Openly available. Indeed, in section 4.3.3 I described a method to automate the use of CheckCIF, which could be used to validate thousands of CIF files per day, many times more than is currently produced (whether published or not).

In order to make crystallographic data visible on the Web, we must associate it with chemical identifiers by calculating the connection tables of the moieties contained in each structure. Unfortunately, this process must

contain heuristics and so is not error-free. In the case of CrystalEye, the molecular skeletons can be unambiguously derived (though sometimes the disorder information may not be resolved), though it is the addition of bond orders and charges which requires heuristics, and therefore poses the toughest problem. Thus, despite there being an Open method to validate the crystallographic data, we risk introducing errors to the data when deriving the chemistry contained within. I believe that introducing any errors into data is unacceptable, and so it is imperative that rigorous checks are used during the heuristic process. If any ambiguities are discovered in the way the data can be interpreted, then it is better to do nothing at all that risk introducing errors. Of course, this is where those secondary data providers, such as the CCDC[131], excel. They use trained chemists to aggregate and interpret data, and have no doubt spent tens, if not hundreds of thousands of man-hours in creating the CSD. However, I believe that most of the process of creating such a database can be automated, as described throughout this work. If the software can be written in such a way that those parts which execute with a high degree of certainty are automated, but ambiguous data is delegated to a human to interpret, then this would be an optimal system. This is the vision for the future of CrystalEye, though as yet a system for the delegation of problem structures to humans for validation has yet to be implemented. We have, however, made a start in creating an editor, C3DE[237] (see figure 7.1), which allows users to edit the bond orders and charges on connection tables generated from a CIF file. Ideally we would have CrystalEye create a webpage containing a C3DE applet, which listed all structures that could not be processed completely. A user would then be able to click through each structure, add the correct bond orders and charges, and save the data.

Many parties are naturally skeptical about the quality of unpublished Open Data (*e.g.* self-published *via* Open Notebook Science, or deposited into an Institutional Repository), however the amount of unpublished data available will soon dwarf all of the data made available in the history of research. It is not going to be a question of whether or not to use unpublished data, but



which unpublished data to use. I think that a lot of responsibility for the future of data quality will lie in data aggregators such as CrystalEye. By providing their data Openly on clear webpages with suitable visualizations, they will provide rallying points for those in the community eager to help and make it simple for them to do so (as with our Open NMR experiment using NMRShiftDB data in chapter 6). It is also expected that federations such as that provided by the upcoming eCrystals project will also be beneficial, as if each repository has access to all the others data, with each using their own validation methods, then the data will be subject to more rigorous tests than if a single 'silo' repository was used.

# Appendix A

## Analysis of the efficacy of Web search engines for chemical search

The following is the supplemental information provided with the paper *Enhancement of the Chemical Semantic Web through InChIfication*[239]. The results and discussion within were performed and written by the author.

### A.1 Search-engines and strategy

To examine the ability of today's search engines at indexing and returning both CAS numbers and InChIs, searches were performed on the search engines shown in table A.1.

Name	URL
Google	<a href="http://www.google.com">http://www.google.com</a>
AOL Search	<a href="http://search.aol.com">http://search.aol.com</a>
Yahoo	<a href="http://www.yahoo.com">http://www.yahoo.com</a>
Altavista	<a href="http://www.altavista.com">http://www.altavista.com</a>
MSN Search	<a href="http://search.msn.com">http://search.msn.com</a>
Ask Jeeves	<a href="http://www.ask.com">http://www.ask.com</a>
Teoma	<a href="http://www.teoma.com">http://www.teoma.com</a>
Dogpile	<a href="http://www.dogpile.com">http://www.dogpile.com</a>

Table A.1: Selection of search engines used in the analysis

This covers the three most popular search engine ‘providers’ (of free listings) in Google, Yahoo and Teoma and also others for which they provide main listings (Google  $\leftarrow$  AOL Search, Yahoo  $\leftarrow$  Altavista / MSN Search, Teoma  $\leftarrow$  Ask Jeeves). Also included was Dogpile[240], a popular meta search engine that draws results from a number of other search engines (including Google, Yahoo, Ask Jeeves and Overture) and returns the ones it considers relevant. We believe that engines in different countries may give different results but this will probably not affect our subject matter and this was not controlled for.

It is extremely important to realise that the analysis of search engines is an inexact science. Search engines do not describe their indexing and retrieval methods (presumably to protect competitive advantages) and may change their strategies at frequent intervals to deter manipulations of rankings. The number of indexed pages changes every minute and a search is, ipso facto, not reproducible. We therefore quote approximate times that searches were performed and many of our conclusions should be adjusted for this. However the results we show have such clear outcomes that we are confident that variations in time and place do not affect them. In the following discussion all searches are in **bold type**.

It is not clear which pages are indexed by search engines but we believe the following:

- Only static web pages are indexed. Pages created on-the-fly as a result of database queries seem to be ignored.
- All engines seem to index X/HTML, DOC, PDF, TXT but may not always index XML (or CML) files. Searching is performed on the ASCII content (XML markup and formatting is ignored).
- All pages fitting the selection above are indexed; no documents are rejected because they are meaningless to humans. There is no reason why any chemical documents should not be included.

By default most engines appear to use the following strategy:

- Use case-insensitive strings, we believe this cannot be altered. This is unlikely to cause many problems in practice; although, for example, CO and Co would collide there is normally enough syntactic context to remove ambiguity.
- Do not carry out stemming and other lexical operations in the presumed language of the query. Thus Google-2004-11-21 retrieves 77 documents for **datument** and 111 for **datuments**. There is apparently no commonality (unless the documents contain both words).



- Use a list of stop words. If Google-2004-11 is given **to be or not to be, that is the question**. It returns:

*The following words are very common and were not included in your search: to **be to be that is the**. Lowercase “or” was ignored. Try “**OR**” to search for either of two terms*

and searches only for **not** and **question**. This gives many false positives.

- By default search engines now seem to apply AND operations between multiple terms so that **wallaby datument** returns no hits on any engines (2004-11-21). This appears to be a change in policy from earlier versions which allowed some OR results to be returned with low numbers of hits.
- Strings can be found anywhere in the document. Thus **chemical semantic web** returns ca. 70, 000 hits, most of which contain the three tokens in widely separated positions. We believe, however, that some engines will order results to emphasize proximity.
- Tokenize the query at any punctuation, Thus Altavista-2004-11-21 returned 298 results for **murray-rust wallaby**. A typical one included the summary:

*...Dance of Death (album)David **Murray**. Enhanced CD  
... British Heavy Metal. Power metal. **Rust**. Slayer ...  
banded gecko. Banded Hare-**Wallaby** (enc.) ...*

Note that the search terms are not proximal and can be de-hyphenated or re-hyphenated.

- Hide or coalesce "similar" results. Thus Google-2004-11-21 returns 3 results for "114795-97-0" but adds the message:

*In order to show you the most relevant results, we have omitted some entries very similar to the 3 already displayed. If you like, you can repeat the search with the omitted results included.*

Following this link now gives a total of 4 hits. This strategy was essential in analysing the recall and precision of InChIs, as different InChIs can appear "similar" to search engines.

All search engines appear to allow refinements of the query to increase precision, and seem to use a common syntax and behaviour. Some options can be provided by syntax but most are on a special "Advanced Search" page.

- Tokens can be mandatory (prefixed by +) or disallowed (-). Thus **+chemical +semantic +web** would recall only those pages which contain all three words. However search engines frequently seem to adopt this strategy by default and the OR option needs to be switched on.
- Enclosing strings in quotes requires that all tokens be returned in the precise order and juxtaposed. Thus **semantic chemical web** would recall **chemical semantic web** but **"semantic chemical web"** would not. However any punctuation appears to be normalised to white-space, so that **"(chemical) semantic-web"** returns the same hits as **"chemical semantic web"**.
- Strings can be long; we have not ascertained a maximum length but Google-2004-11-21 retrieves a 60-character protein sequence fragment such as

EPTTMITLGPLLVLVIGFFAWLLFTLAVFALPVFAGVTIGLWAFHTGAGALGGIAVGPV

while a substring, as expected, is not retrieved.

Search engines seem to have a maximum number of tokens in a quoted string; apparently 10 in Google-2004-11 and AltaVista-2004-11. A Google-2004-11-21 search for **"twas brillig and the slithy toves did gyre and gimble in the wabe"** returns 4020 hits and advises

*"in" (and any subsequent words) was ignored because we limit queries to 10 words.*

So the same hits are returned for the string **"twas brillig and the slithy toves did gyre and gimble in the bath"**. Initially this is counted as false positive, but it is easily eliminated by a textual search of the document. This means that a simple tool can match the full text of the retrieved documents and eliminate hits that did not contain the full search string. This is important for InChIs which almost always have more than 10 tokens.

Search Engine	No. of entries viewable
Google	1000
AOL Search	535
Yahoo	1000
Altavista	1050
MSN Search	1000
Ask Jeeves	200
Teoma	200
Dogpile	125

Table A.2: Number of query results viewable for a selection of commercial Web search engines

Search engines also seem to have a limit to the number of the recalled entries you can view. On searching for a common word you may be informed that there are 10,000 documents found that match, but you will never be able to view them all. table A.2 shows the maximum number of results that can be viewed for each search engine (2004-11-21). This could prove a problem for the future, particularly if someone were to search for a large molecule which had more stereoisomers than viewable results from the search engine.

## A.2 Search terms and metrics

In information retrieval (IR) it is normal practice to define a precise corpus which is to be searched (or processed) and to measure the recall and precision of different strategies. In this case the unit of measurement is the page (or document). It is possible that a page may contain multiple instances of search terms but this was not relevant here. We use the abbreviations and terms:

$P$  total known pages

$H$  total pages retrieved (hits)

$TP$  true positive; all those retrieved pages which match the InChI

$FP1$  false positive matching another InChI

$FP2$  false positive matching a non-InChI string

$FP$  total false positive

$FN$  false negative; pages containing the query InChI which were not retrieved

Note that:

- $FP = FP1 + FP2$
- $H = TP + FP$
- $P = TP + FN$
- $recall = TP / P$
- $precision = TP / H$

These concepts are not applicable when the size of the corpus is unknown, as for the searches for CAS registry numbers.

### A.3 Searching for CAS numbers

Chemical Abstracts registry numbers ("CAS numbers") are widely used across the World Wide Web and can also act as unique identifiers. To test precision and recall two very common compounds, caffeine and acetic acid were chosen. These occur in many types of document (journal articles, suppliers' catalogs, Materials Safety Data Sheets, lists of properties) and in multiple sources. The precise number of web pages containing a given CAS number is unknown and changes continuously, so that recall cannot be established. (As CAS numbers are copyright, we assumed it may not be legal to create test documents without permission). In practice authors of web pages use a variety of syntaxes such as **CAS: 64-19-7**, **CAS number: 64-19-7**, **Registry number: 64-19-7** and frequently simply **64-19-7** (almost universal when tables are used). Recall will obviously be higher for the pure number but precision will be lower.

It is impossible to measure the precision and recall of the whole corpus (often  $>10^4$  entries). The first 100 hits with 3 strategies were therefore analysed (table A.3). Note that the total number of hits varies enormously and that the aggregator (Dogpile) is clearly selective. MSN Search seems to select on single tokens and neglects order. The contrast in precision between caffeine and acetic acid (Yahoo, MSN Search, Teoma) is surprising since the actual tokens are probably relatively equifrequent. Some differences may be due in part to speed of indexing. It is interesting that adding the apparent constraint **+number** to the search actually increases the total hits.

true positives / sample size (total hits)

Search string	Google	AOL Search	Yahoo	Altavista	MSN Search	Ask Jeeves	Teoma	Dogpile
	Acetic acid: 64-19-7							
"64-19-7"	79/100(15,100)	78/100(3220)	9/100(10,200)	27/100(9,830)	63/100(1,486)	82/100(2,950)	82/100(2,950)	28/48(48)
+CAS +"64-19-7"	100/100(10,800)	99/100(2,230)	99/100(4,400)	99/100(4,250)	100/100(881)	100/100(1,530)	100/100(1,530)	62/63(63)
+CAS +number +"64-19-7"	100/100(5,300)	100/100(1125)	100/100(1,380)	100/100(1,500)	99/100(495)	100/100(1,060)	100/100(1,060)	60/61(61)
	Caffeine: 58-08-2							
"58-08-2"	28/100(6,720)	28/100(1,435)	0/100(543,000)	0/100(550,000)	0/100(100,488)	43/100(2,540)	43/100(2,540)	9/11(11)
+CAS +"58-08-2"	100/100(873)	100/100(550)	23/100(13,300)	20/100(13,400)	21/100(2,289)	98/100(396)	98/100(396)	21/22(22)
+CAS +"number" +"58-08-2"	100/100(8,250)	100/100(265)	32/100(4,430)	34/100(2,080)	25/100(601)	94/100(207)	94/100(207)	30/56(56)

Table A.3: Searching for CAS numbers with various strings(2004-11-18)

Manual examination of the first 100 entries showed the number of false positives (i.e. not chemical compounds). Not surprisingly this can be a high percentage for the raw strings as they retrieve many other triads (dates, phone numbers, etc.) and the enormous amount of noise for **58-08-02** seems to be due in part to ringtones. It seems that the string CAS provides complete recall in some cases but at the expense of precision (<50%).

## A.4 Searching for InChIs

### A.4.1 The InChI architecture and implications

An InChI string consists of layers, the first being the chemical formula of the compound. This is followed by the atom connection information, which in turn is followed by optional layers containing information such as stereochemistry or isotopic content. For most molecules there are many more than 10\* tokens that appear before the connection information is complete. We can see that when searching for a stereoisomer of a large molecule, that none of the information in the stereochemical layer of the InChI string will be included in a search by Google. Thus if InChI strings of other stereoisomers were on the web, then they would be seen as identical to the search string and incorrectly returned. The same can be said of any molecules with the same connection information but differing information in the later layers. Indeed, it would also be possible for two different molecules to have the same chemical formula and also start of the connection information but have differing connection information in the regions not searched for by Google. So unless search engines start searching with the whole search string entered, this could cause minor problems in the future. If not, a program used post-search to scan the recalled entries for the complete InChI string would be necessary to ensure no ‘other-InChI’ false positives.

---

\*Since these searches were performed (2004-11), Google has increased the maximum number of tokens used for a query from 10 to 32 (2005-02). Thus the same hits are now not returned for **"twas brillig and the slithy toves did gyre and gimble in the wabe"** and **"twas brillig and the slithy toves did gyre and gimble in the bath"**. However, **"twas brillig, and the slithy toves did gyre and gimble in the wabe all mimsy were the borogoves, and the mome raths outgrabe. Beware the Jabberwock, my son! The jaws that bite, the claws that catch!"** and **"twas brillig, and the slithy toves did gyre and gimble in the wabe all mimsy were the borogoves, and the mome raths outgrabe. Beware the Jabberwock, my son! The jaws that bite, the rods that fish!"** do return the same hits.

## A.4.2 Results

There are very few InChIs on the web, so our experiment was performed on a bounded dataset of unique compounds. We chose the University of Southampton’s Crystal Structure Report Archive website[241], which at the time (2004-11-18) contained 104 pages each containing the results of a crystal structure. Each page contains an IChI (sic the name started out as IChI and was changed then to INChI before finally settling on InChI) string, created with version 0.932Beta of the identifier. Note that the compounds are mainly novel and/or complex so it is extremely unlikely that anyone outside the authors will have published the same compounds using IChIs in a different context, especially as V0.932 beta is now obsolete. We therefore have an accurate estimate of recall as  $\text{totalHits} / \text{totalKnownPages}$ . This corpus consists of:

- 104 HTML pages
- 100 describing different molecules for which the IChI is unique
- 2 pages describing different experiments on the same compound and will therefore have the same IChI
- 2 pages describing a pair of diastereomers which have the same IChI as stereochemistry was omitted from its calculation. This is an inter-IChI collision which the search engines obviously cannot detect.

There are thus a total of 102 IChIs on the 104 Southampton HTML pages.

- 93 CML pages which are mapped to the HTML pages
- 89 different molecules with unique IChIs
- 2 of the two experiments on the same molecule
- 2 of the two diastereomers

There are thus a total of 91 IChIs on the 93 Southampton CML pages. To simplify the analysis we report the HTML and CML retrieval separately. For each of the 102 IChIs a separate search was performed on each engine, using the quoted IChI string. The results are aggregated in table A.4. The results of the search for each engine are aggregated within cells (described in the caption). The search was performed on two dates and shows a significant increase in the MSN recall.

Out of the 832 searches performed on 8 different search engines there were no false positives.

TP / FP1 / FP2 / P / recall(%) / precision(%)

	2004-11-05		2004-11-18	
	.html	.cml	.html	.cml
<i>Pages</i>	<i>104</i>	<i>93</i>	<i>104</i>	<i>93</i>
Google	103/0/0/99/100	67/0/0/72/100	104/0/0/100/100	92/0/0/99/100
AOL Search	N/A		102/0/0/98/100	91/0/0/98/100
Yahoo	15/0/0/14/100	0/0/0/0/-	33/0/0/32/100	0/0/0/0/-
Altavista	20/0/0/19/100	0/0/0/0/-	39/0/0/38/100	0/0/0/0/-
MSN Search	0/0/0/0/-	0/0/0/0/-	43/0/0/42/100	0/0/0/0/-
Ask Jeeves	N/A		0/0/0/0/-	0/0/0/0/-
Teoma	N/A		0/0/0/0/-	0/0/0/0/-
Dogpile	N/A		102/0/0/98/100	91/0/0/98/100

Table A.4: Recall of InChI strings from the Crystal Structure Report Archive (2004-11-18)

### Why is there a difference in recall between InChI strings and CAS numbers?

A CAS number consists of only numbers separated by generic punctuation and an InChI string consists of blocks of letters and numbers separated by generic punctuation. CAS numbers are at a disadvantage as they are short and only contain numbers and generic punctuation. Indeed, when searching for the CAS number of caffeine **58-08-2** it is quite common for documents containing information on acetone to be recalled, as its molecular weight is **58.08**. InChI strings are generally much longer than CAS numbers and as they have a good mix of letters and numbers in their tokens and separation by generic punctuation, it is unlikely that even a small section of an InChI string will be matched to anything else on the World Wide Web.

## A.5 Searching for SMILES

In principle SMILES is uniqueifiable but in practice there is no public conformance to the specification and we also believe that some implementations produce incompatible results. The present study confirms the variation. To determine the usage of SMILES, queries of the form **SMILES AND ("caffeine" OR "58-08-2")** were used. This is clearly imprecise, but about 10 sites were found containing SMILES for caffeine which showed at least 7 different syntactic variants. To test precision and recall these were submitted to Google-2004-11-20.

The raw precision of the SMILES strings in table A.5 is high. This can be attributed to each string (apart from the first) having around 10 tokens, allowing Google to search for the whole string and match it exactly with strings found on the web. The first string has 18 tokens, and so a large



True positives/False positives/non-SMILES(Total hits)

Search String	Google	Located on sites
"[c]1([n+](CH3)[c]([c]2([c]([n+]1CH3)[n][cH][n+]2CH3)))[0-][0-]"	2/3/0(5)	www.biocheminfo.org www.eureka.ya.com www.biozentrum.unibas.ch
"CN1C(=O)N(C)C(=O)C(N(C)C=N2)=C12"	14/0/0(14)	www.daylight.com
"Cn1cnc2n(C)c(=O)n(C)c(=O)c12"	20/0/0(20)	www.daylight.com pubs.acs.org www.predictive-toxicology.org www.eyesopen.com bind.ca www.surrey.ac.uk www.sunsetmolecular.com
"Cn1cnc2c1c(=O)n(C)c(=O)n2C"	2/0/0(2)	www.molinspiration.com doi.wiley.com
"N1(C)C(=O)N(C)C2=C(C1=O)N(C)C=N2"	1/0/0(1)	www.fda.gov
"O=C1C2=C(N=CN2C)N(C(=O)N1C)C"	2/0/0(2)	potency.berkeley.edu
"CN1C=NC2=C1C(=O)N(C)C(=O)N2C"	17/0/0(17)	www.jchem.com www.chemaxon.com www.structuresearch.com www.chemaxon.hu bohlmann.bgbm.org

Table A.5: Searching for SMILES representations of caffeine in Web search engines

part is not included in the search, leading to lower precision. To a search engine SMILES strings are similar to InChI strings as they consist of tokens containing letters and numbers separated by generic punctuation.

The 58 pages from table A.5 occurred on 20 sites with 7 syntactic variants. There is thus no commonality of approach (i.e. the page creators are not using a synoptic approach).

## A.6 Searching for InChI strings from the KEGG collection using Google

On 2004-10-04 9585 molecules from the KEGG collection were converted to CML, indexed with InChI V1.12 Beta and posted as static pages on the WWW. To our knowledge there are very few other InChI V1.12 Beta instances on the web, so this provides a test of recall.

At 2004-11-16 the molecules at [wwmm.ch.cam.ac.uk/data/kegg](http://wwmm.ch.cam.ac.uk/data/kegg) have been indexed on Google up to c07576. As far as we know the indexing is serial so that 4870 molecules were indexed. To test precision and recall the InChIs for 83 KEGG ligands c00001-c00100 were submitted to Google (table A.6).

There were no non-InChI recalls (FP1 = 0) and no false negatives (FN = 0). No hits were found to molecules not on our site so recall is measured with respect to this. The FP2 are due to collisions after the first 10 unique tokens. They are easily removed by a simple program filtering the search engine results. For each false positive its quoted InChI was submitted to see if all the recalls were symmetric (i.e. if molecule c00010 recalls c00298, does c00298 recall c00010). There are 35 isomeric collisions in about 4500 molecules, which suggests that the false collisions can be managed with simple filters. In practice, also, many molecules in KEGG do not have complete stereochemistry so that methods other than the connection table (e.g. names) would have to be used to separate them.

### Isomers with InChI collisions

Each of the 35 entries contains two or more colliding InChIs, for each of which the serial number (not the KEGG id) and the InChI are given. The entry is optionally followed by a note. Note that all InChI collisions are trivially resolvable after retrieval and are listed to give an idea of the length of the strings that most search engines index. The only problem arises if collisions are so frequent that the search engine cuts off before returning all the true positives.

Table A.6: Collisions in web queries for InChI representations of Kegg molecules

Kegg No.	InChI
c01328 c00001	1.12Beta/H2O/h1H2/p-1 1.12Beta/H2O/h1H2 <i>NOTE: 1 (water) recalls 1328 BUT 1328 does not recall 1</i>
c00704 c00007	1.12Beta/O2/C1-2 1.12Beta/O2/C1-2 <i>NOTE: 704 is the superoxide ion O2- and should have been rendered as 1.12Beta/O2/C1-2/q-1</i>
c00054  c00008  c03850	1.12Beta/C10H15N5O10P2/C11-8-5-9(13-2-12-8)15(3-14-5)10-6(16)7(25-27(20,21)22)4(24-10)1-23-26(17,18)19/h1H2,2-4H,6-7H,10H,16H,(H2,11,12,13)(H2,17,18,19)(H2,20,21,22)/t4-,6-,7-,10-/m1/s1 1.12Beta/C10H15N5O10P2/C11-8-5-9(13-2-12-8)15(3-14-5)10-7(17)6(16)4(24-10)1-23-27(21,22)25-26(18,19)20/h1H2,2-4H,6-7H,10H,16-17H,(H,21,22)(H2,11,12,13)(H2,18,19,20)/t4-,6-,7-,10-/m1/s1 1.12Beta/C10H15N5O10P2/C11-8-5-9(13-2-12-8)15(3-14-5)10-7(25-27(20,21)22)6(16)4(24-10)1-23-26(17,18)19/h1H2,2-4H,6-7H,10H,16H,(H2,11,12,13)(H2,17,18,19)(H2,20,21,22)/t4-,6-,7-,10-/m1/s1 <i>NOTE: These are isomers but the connection table only differs after the 10th token</i>
c00014 c01342	1.12Beta/H3N/h1H3 1.12Beta/H3N/h1H3/p+1 <i>NOTE: 14 recalls 1342 BUT 1342 does not recall 14</i>

Continued on next page

Table A.6 – continued from previous page

Kegg No.	InChI
c01367	1.12Beta/C10H14N5O7P/C11-8-5-9(13-2-12-8)15(3-14-5)10-6(17)7(4(1-16)21-10)22-23(18,19)20/h1H2,2-4H,6-7H,10H,16-17H,(H2,11,12,13)(H2,18,19,20)/t4-,6-,7-,10-/m1/s1
c04378	1.12Beta/C10H14N5O7P/C11-8-5-9(13-2-12-8)14-3-15(5)10-7(17)6(16)4(22-10)1-21-23(18,19)20/h1H2,2-4H,6-7H,10H,16-17H,(H2,11,12,13)(H2,18,19,20)/t4-,6-,7-,10-/m1/s1
c00946	1.12Beta/C10H14N5O7P/C11-8-5-9(13-2-12-8)15(3-14-5)10-7(22-23(18,19)20)6(17)4(1-16)21-10/h1H2,2-4H,6-7H,10H,16-17H,(H2,11,12,13)(H2,18,19,20)/t4-,6-,7-,10-/m1/s1
c00020	1.12Beta/C10H14N5O7P/C11-8-5-9(13-2-12-8)15(3-14-5)10-7(17)6(16)4(22-10)1-21-23(18,19)20/h1H2,2-4H,6-7H,10H,16-17H,(H2,11,12,13)(H2,18,19,20)/t4-,6-,7-,10-/m1/s1
c00023	1.12Beta/Fe
c00824	1.12Beta/Fe.H2S/h;1H2/q+1;/p-1 NOTE: 23 recalls 824 BUT 824 does not recall 23 NOTE: c00023 is junk as no proper charge is given
c00217	1.12Beta/C5H9NO4/c6-3(5(9)10)1-2-4(7)8/h1-2H2,3H,6H2,(H,7,8)(H,9,10)/t3-/m1/s1
c00302	1.12Beta/C5H9NO4/c6-3(5(9)10)1-2-4(7)8/h1-2H2,3H,6H2,(H,7,8)(H,9,10)
c00025	1.12Beta/C5H9NO4/c6-3(5(9)10)1-2-4(7)8/h1-2H2,3H,6H2,(H,7,8)(H,9,10)/t3-/m0/s1
c00029	1.12Beta/C15H24N2O17P2/C18-3-5-8(20)10(22)12(24)14(32-5)33-36(28,29)34-35(26,27)30-4-6-9(21)11(23)13(31-6)17-2-1-7(19)16-15(17)25/h1-2H,3-4H2,5-6H,8-14H,18H,20-24H,(H,26,27)(H,28,29)(H,16,19,25)/t5-,6-,8+,9-,10+,11-,12+,13-,14?/m1/s1
c00052	1.12Beta/C15H24N2O17P2/C18-3-5-8(20)10(22)12(24)14(32-5)33-36(28,29)34-35(26,27)30-4-6-9(21)11(23)13(31-6)17-2-1-7(19)16-15(17)25/h1-2H,3-4H2,5-6H,8-14H,18H,20-24H,(H,26,27)(H,28,29)(H,16,19,25)/t5-,6-,8+,9-,10-,11-,12-,13?,14?/m1/s1
c00936	1.12Beta/C6H12O6/c7-1-2-3(8)4(9)5(10)6(11)12-2/h1H2,2-11H/t2?,3?,4-,5-,6-/m0/s1
c00124	1.12Beta/C6H12O6/c7-1-2-3(8)4(9)5(10)6(11)12-2/h1H2,2-11H/t2-,3+,4-,5+,6?/m0/s1
c00159	1.12Beta/C6H12O6/c7-1-2-3(8)4(9)5(10)6(11)12-2/h1H2,2-11H/t2-,3-,4-,5-,6?/m0/s1
c00031	1.12Beta/C6H12O6/c7-1-2-3(8)4(9)5(10)6(11)12-2/h1H2,2-11H/t2-,3-,4-,5+,6?/m0/s1
c06467	1.12Beta/C6H12O6/c7-1-2-3(8)4(9)5(10)6(11)12-2/h1H2,2-11H/t2-,3+,4-,5+,6?/m1/s1
c06464	1.12Beta/C6H12O6/c7-1-2-3(8)4(9)5(10)6(11)12-2/h1H2,2-11H/t2-,3-,4+,5+,6?/m1/s1
c01487	1.12Beta/C6H12O6/c7-1-2-3(8)4(9)5(10)6(11)12-2/h1H2,2-11H/t2-,3-,4+,5-,6?/m1/s1
c00221	1.12Beta/C6H12O6/c7-1-2-3(8)4(9)5(10)6(11)12-2/h1H2,2-11H/t2?,3?,4-,5+,6+/m0/s1
c00267	1.12Beta/C6H12O6/c7-1-2-3(8)4(9)5(10)6(11)12-2/h1H2,2-11H/t2-,3-,4-,5+,6-/m0/s1
c00962	1.12Beta/C6H12O6/c7-1-2-3(8)4(9)5(10)6(11)12-2/h1H2,2-11H/t2?,3?,4-,5+,6+/m0/s1

Continued on next page

Table A.6 – continued from previous page

Kegg No.	InChI
c00984	1.12Beta/C6H12O6/c7-1-2-3(8)4(9)5(10)6(11)12-2/h1H2, 2-11H/t2-, 3+, 4-, 5+, 6-/m0/s1
c06465	1.12Beta/C6H12O6/c7-1-2-3(8)4(9)5(10)6(11)12-2/h1H2, 2-11H/t2-, 3+, 4+, 5-, 6?/m1/s1
c06466	1.12Beta/C6H12O6/c7-1-2-3(8)4(9)5(10)6(11)12-2/h1H2, 2-11H/t2-, 3+, 4+, 5+, 6?/m1/s1
c00293	1.12Beta/C6H12O6/c7-1-2-3(8)4(9)5(10)6(11)12-2/h1H2, 2-11H/t2-, 3-, 4+, 5-, 6?/m1/s1
c01582	1.12Beta/C6H12O6/c7-1-2-3(8)4(9)5(10)6(11)12-2/h1H2, 2-11H/t2-, 3+, 4+, 5-, 6+/m1/s1
c01825	1.12Beta/C6H12O6/c7-1-2-3(8)4(9)5(10)6(11)12-2/h1H2, 2-11H/t2-, 3+, 4+, 5-, 6+/m0/s1
c02209	1.12Beta/C6H12O6/c7-1-2-3(8)4(9)5(10)6(11)12-2/h1H2, 2-11H/t2-, 3-, 4-, 5-, 6+/m0/s1
c00738	1.12Beta/C6H12O6/c7-1-2-3(8)4(9)5(10)6(11)12-2/h1H2, 2-11H
c01381	1.12Beta/C6H12O6/c7-1-2-3(8)4(9)5(10)6(11)12-2/h1H2, 2-11H NOTE: these hexopyranosides (stereoisomers of glucose) show the variation in stereochemical information in KEGG: <ul style="list-style-type: none"> <li>• c02209: exact at all centres</li> <li>• c00738: no stereochemical information given</li> <li>• c00936: stereochemistry given at 3/5 centres</li> </ul>
c02606	1.12Beta/C4H4O5/c5-2(4(8)9)1-3(6)7/h1H, 5H, (H, 6, 7) (H, 8, 9)/b2-1-
c03981	1.12Beta/C4H4O5/c5-2(4(8)9)1-3(6)7/h1H, 5H, (H, 6, 7) (H, 8, 9)/b2-1-
c00036	1.12Beta/C4H4O5/c5-2(4(8)9)1-3(6)7/h1H2, (H, 6, 7) (H, 8, 9)
c00133	1.12Beta/C3H7NO2/C1-2(4)3(5)6/h1H3, 2H, 4H2, (H, 5, 6)/t2-/m1/s1
c01401	1.12Beta/C3H7NO2/C1-2(4)3(5)6/h1H3, 2H, 4H2, (H, 5, 6)
c00041	1.12Beta/C3H7NO2/C1-2(4)3(5)6/h1H3, 2H, 4H2, (H, 5, 6)/t2-/m0/s1
c00203	1.12Beta/C17H27N3O17P2/C1-6(22)18-10-13(26)11(24)7(4-21)35-16(10)36-39(31, 32)37-38(29, 30)33-5-8-12(25)14(27)15(34-8)20-3-2-9(23)19-17(20)28/h1H3, 2-3H, 4-5H2, 7-8H, 10-16H, 21H, 24-27H, (H, 18, 22) (H, 29, 30) (H, 31, 32) (H, 19, 23, 28)/t7-, 8-, 10-, 11+, 12-, 13-, 14-, 15?, 16?/m1/s1
c01170	1.12Beta/C17H27N3O17P2/C1-6(22)18-10-13(26)11(24)7(4-21)35-16(10)36-39(31, 32)37-38(29, 30)33-5-8-12(25)14(27)15(34-8)20-3-2-9(23)19-17(20)28/h1H3, 2-3H, 4-5H2, 7-8H, 10-16H, 21H, 24-27H, (H, 18, 22) (H, 29, 30) (H, 31, 32) (H, 19, 23, 28)/t7-, 8-, 10-, 11+, 12-, 13-, 14-, 15-, 16?/m1/s1
c00043	1.12Beta/C17H27N3O17P2/C1-6(22)18-10-13(26)11(24)7(4-21)35-16(10)36-39(31, 32)37-38(29, 30)33-5-8-12(25)14(27)15(34-8)20-3-2-9(23)19-17(20)28/h1H3, 2-3H, 4-5H2, 7-8H, 10-16H, 21H, 24-27H, (H, 18, 22) (H, 29, 30) (H, 31, 32) (H, 19, 23, 28)/t7-, 8-, 10-, 11-, 12-, 13+, 14-, 15-, 16?/m1/s1
c00047	1.12Beta/C6H14N2O2/c7-4-2-1-3-5(8)6(9)10/h1-4H2, 5H, 7-8H2, (H, 9, 10)/t5-/m0/s1

Continued on next page

Table A.6 – continued from previous page

Kegg No.	InChI
c00739	1.12Beta/C6H14N2O2/c7-4-2-1-3-5(8)6(9)10/h1-4H2,5H,7-8H2,(H,9,10)/t5-/m1/s1
c00049	1.12Beta/C4H7NO4/c5-2(4(8)9)1-3(6)7/h1H2,2H,5H2,(H,6,7)(H,8,9)/t2-/m1/s1
c00402	1.12Beta/C4H7NO4/c5-2(4(8)9)1-3(6)7/h1H2,2H,5H2,(H,6,7)(H,8,9)/t2-/m1/s1
c00055	1.12Beta/C9H14N3O8P/C10-5-1-2-12(9(15)11-5)8-7(14)6(13)4(20-8)3-19-21(16,17)18/h1-2H,3H2,4H,6-8H,13-14H,(H2,10,11,15)(H2,16,17,18)/t4-,6-,7-,8-/m1/s1
c05822	1.12Beta/C9H14N3O8P/C10-5-1-2-12(9(15)11-5)8-6(14)7(4(3-13)19-8)20-21(16,17)18/h1-2H,3H2,4H,6-8H,13-14H,(H2,10,11,15)(H2,16,17,18)/t4-,6-,7-,8-/m1/s1
c03104	1.12Beta/C9H14N3O8P/C10-5-1-2-12(9(15)11-5)8-7(20-21(16,17)18)6(14)4(3-13)19-8/h1-2H,3H2,4H,6-8H,13-14H,(H2,10,11,15)(H2,16,17,18)/t4-,6-,7-,8-/m1/s1
c00062	1.12Beta/C6H15N4O2/c7-4(5(11)12)2-1-3-10-6(8)9/h1-3H2,4H,7-9H2,10H,(H,11,12)/t4-/m0/s1
c00792	1.12Beta/C6H15N4O2/c7-4(5(11)12)2-1-3-10-6(8)9/h1-3H2,4H,7-9H2,10H,(H,11,12)/t4-/m1/s1
c02385	1.12Beta/C6H15N4O2/c7-4(5(11)12)2-1-3-10-6(8)9/h1-3H2,4H,7-9H2,10H,(H,11,12)
c00064	1.12Beta/C5H10N2O3/c6-3(5(9)10)1-2-4(7)8/h1-2H2,3H,6H2,(H2,7,8)(H,9,10)/t3-/m0/s1
c00819	1.12Beta/C5H10N2O3/c6-3(5(9)10)1-2-4(7)8/h1-2H2,3H,6H2,(H2,7,8)(H,9,10)/t3-/m1/s1
c00303	1.12Beta/C5H10N2O3/c6-3(5(9)10)1-2-4(7)8/h1-2H2,3H,6H2,(H2,7,8)(H,9,10)
c00716	1.12Beta/C3H7NO3/c4-2(1-5)3(6)7/h1H2,2H,4H2,5H,(H,6,7)
c00065	1.12Beta/C3H7NO3/c4-2(1-5)3(6)7/h1H2,2H,4H2,5H,(H,6,7)/t2-/m0/s1
c00740	1.12Beta/C3H7NO3/c4-2(1-5)3(6)7/h1H2,2H,4H2,5H,(H,6,7)/t2-/m1/s1
c00072	1.12Beta/C6H8O6/c7-1-2(8)5-3(9)4(10)6(11)12-5/h1H2,2H,5H,7-10H/t2-,5+/m0/s1
c06430	1.12Beta/C6H8O6/c7-1-2(8)5-3(9)4(10)6(11)12-5/h1-5H,8-10H/t2-,3+,4+,5-/m1/s1
c03289	1.12Beta/C6H8O6/c7-1-2(8)5-3(9)4(10)6(11)12-5/h1H2,2-3H,5H,7-9H/t2-,3+,5+/m0/s1
c01733	1.12Beta/C5H11NO2S/C1-9-3-2-4(6)5(7)8/h1H3,2-3H2,4H,6H2,(H,7,8)
c00855	1.12Beta/C5H11NO2S/C1-9-3-2-4(6)5(7)8/h1H3,2-3H2,4H,6H2,(H,7,8)/t4-/m1/s1
c00073	1.12Beta/C5H11NO2S/C1-9-3-2-4(6)5(7)8/h1H3,2-3H2,4H,6H2,(H,7,8)/t4-/m0/s1
c01602	1.12Beta/C5H12N2O2/c6-3-1-2-4(7)5(8)9/h1-3H2,4H,6-7H2,(H,8,9)
c00077	1.12Beta/C5H12N2O2/c6-3-1-2-4(7)5(8)9/h1-3H2,4H,6-7H2,(H,8,9)/t4-/m1/s1
c00515	1.12Beta/C5H12N2O2/c6-3-1-2-4(7)5(8)9/h1-3H2,4H,6-7H2,(H,8,9)/t4-/m0/s1
c00806	1.12Beta/C11H12N2O2/C12-9(11(14)15)5-7-6-13-10-4-2-1-3-8(7)10/h1-4H,5H2,6H,9H,12H2,13H,(H,14,15)
c00525	1.12Beta/C11H12N2O2/C12-9(11(14)15)5-7-6-13-10-4-2-1-3-8(7)10/h1-4H,5H2,6H,9H,12H2,13H,(H,14,15)/t9-/m1/s1

Continued on next page

Table A.6 – continued from previous page

Kegg No.	InChI
c00078	1.12Beta/C11H12N2O2/C12-9(11(14)15)5-7-6-13-10-4-2-1-3-8(7)10/h1-4H,5H2,6H,9H,12H2,13H,(H,14,15)/t9-/m0/s1
c00079	1.12Beta/C9H11N02/C10-8(9(11)12)6-7-4-2-1-3-5-7/h1-5H,6H2,8H,10H2,(H,11,12)/t8-/m0/s1
c02265	1.12Beta/C9H11N02/C10-8(9(11)12)6-7-4-2-1-3-5-7/h1-5H,6H2,8H,10H2,(H,11,12)/t8-/m1/s1
c02057	1.12Beta/C9H11N02/C10-8(9(11)12)6-7-4-2-1-3-5-7/h1-5H,6H2,8H,10H2,(H,11,12)
c06420	1.12Beta/C9H11N03/C10-8(9(12)13)5-6-1-3-7(11)4-2-6/h1-4H,5H2,8H,10H2,11H,(H,12,13)/t8-/m1/s1
c01536	1.12Beta/C9H11N03/C10-8(9(12)13)5-6-1-3-7(11)4-2-6/h1-4H,5H2,8H,10H2,11H,(H,12,13)
c00082	1.12Beta/C9H11N03/C10-8(9(12)13)5-6-1-3-7(11)4-2-6/h1-4H,5H2,8H,10H2,11H,(H,12,13)/t8-/m0/s1
c00083	1.12Beta/C24H38N7O19P3S/C1-24(2,19(37)22(38)27-4-3-13(32)26-5-6-54-15(35)7-14(33)34)9-47-53(44,45)50-52(42,43)46-8-12-18(49-51(39,40)41)17(36)23(48-12)31-11-30-16-20(25)28-10-29-21(16)31/h1-2H3,3-9H2,10-12H,17-19H,23H,36-37H,(H,26,32)(H,27,38)(H,33,34)(H,42,43)(H,44,45)(H2,25,28,29)(H2,39,40,41)/t12-,17-,18-,19?,23-/m1/s1
c03188	1.12Beta/C24H38N7O19P3S/C1-24(2,19(37)22(38)27-4-3-13(32)26-5-6-54-15(35)7-14(33)34)9-47-53(44,45)50-52(42,43)46-8-12-18(49-51(39,40)41)17(36)23(48-12)31-11-30-16-20(25)28-10-29-21(16)31/h1-2H3,3-9H2,10-12H,17-19H,23H,36-37H,(H,26,32)(H,27,38)(H,33,34)(H,42,43)(H,44,45)(H2,25,28,29)(H2,39,40,41)/t12-,17-,18-,19?,23-/m1/s1
c00085	1.12Beta/C6H13O9P/c7-2-6(10)5(9)4(8)3(15-6)1-14-16(11,12)13/h1-2H2,3-5H,7-10H,(H2,11,12,13)/t3-,4-,5+,6-/m1/s1
c06312	1.12Beta/C6H13O9P/c7-2-6(10)5(9)4(8)3(15-6)1-14-16(11,12)13/h1-2H2,3-5H,7-10H,(H2,11,12,13)/t3-,4+,5+,6-/m0/s1
c05345	1.12Beta/C6H13O9P/c7-2-6(10)5(9)4(8)3(15-6)1-14-16(11,12)13/h1-2H2,3-5H,7-10H,(H2,11,12,13)/t3-,4-,5+,6-/m1/s1
c01097	1.12Beta/C6H13O9P/c7-2-6(10)5(9)4(8)3(15-6)1-14-16(11,12)13/h1-2H2,3-5H,7-10H,(H2,11,12,13)/t3-,4+,5+,6-/m1/s1
c00283	1.12Beta/H2S/h1H2
c00087	1.12Beta/H2S/h1H2
c00090	1.12Beta/C6H6O2/c7-5-3-1-2-4-6(5)8/h1-4H,7-8H
c05060	1.12Beta/C6H6O2/c7-5-3-1-2-4-6(5)8/h1-5H,7H
c01785	1.12Beta/C6H6O2/c7-5-3-1-2-4-6(5)8/h1-4H,7-8H
c01172	1.12Beta/C6H13O9P/c7-3-2(1-14-16(11,12)13)15-6(10)5(9)4(3)8/h1H2,2-10H,(H2,11,12,13)/t2-,3-,4-,5-,6+/m0/s1
c02962	1.12Beta/C6H13O9P/c7-3-2(1-14-16(11,12)13)15-6(10)5(9)4(3)8/h1H2,2-10H,(H2,11,12,13)
c00275	1.12Beta/C6H13O9P/c7-3-2(1-14-16(11,12)13)15-6(10)5(9)4(3)8/h1H2,2-10H,(H2,11,12,13)/t2-,3-,4+,5+,6-/m1/s1
c00668	1.12Beta/C6H13O9P/c7-3-2(1-14-16(11,12)13)15-6(10)5(9)4(3)8/h1H2,2-10H,(H2,11,12,13)/t2-,3-,4+,5-,6+/m1/s1

Continued on next page



Table A.6 – continued from previous page

Kegg No.	InChI
c01113	1.12Beta/C6H13O9P/c7-3-2(1-14-16(11,12)13)15-6(10)5(9)4(3)8/h1H2,2-10H,(H2,11,12,13)/t2?,3?,4-,5+,6?/m0/s1
c03735	1.12Beta/C6H13O9P/c7-3-2(1-14-16(11,12)13)15-6(10)5(9)4(3)8/h1H2,2-10H,(H2,11,12,13)/t2?,3?,4?,5?,6-/m0/s1
c02965	1.12Beta/C6H13O9P/c7-3-2(1-14-16(11,12)13)15-6(10)5(9)4(3)8/h1H2,2-10H,(H2,11,12,13)
c00092	1.12Beta/C6H13O9P/c7-3-2(1-14-16(11,12)13)15-6(10)5(9)4(3)8/h1H2,2-10H,(H2,11,12,13)/t2-,3-,4+,5-,6?/m1/s1
c00623	1.12Beta/C3H9O6P/c4-1-3(5)2-9-10(6,7)8/h1-2H2,3-5H,(H2,6,7,8)/t3-/m1/s1
c03189	1.12Beta/C3H9O6P/c4-1-3(5)2-9-10(6,7)8/h1-2H2,3-5H,(H2,6,7,8)
c00093	1.12Beta/C3H9O6P/c4-1-3(5)2-9-10(6,7)8/h1-2H2,3-5H,(H2,6,7,8)/t3-/m0/s1
c00095	1.12Beta/C6H12O6/c7-1-3-4(9)5(10)6(11,2-8)12-3/h1-2H2,3-5H,7-11H/t3-,4-,5+,6-/m1/s1
c01719	1.12Beta/C6H12O6/c7-1-3-4(9)5(10)6(11,2-8)12-3/h1-2H2,3-5H,7-11H/t3-,4-,5+,6-/m0/s1
c02336	1.12Beta/C6H12O6/c7-1-3-4(9)5(10)6(11,2-8)12-3/h1-2H2,3-5H,7-11H/t3-,4-,5+,6-/m1/s1
c01496	1.12Beta/C6H12O6/c7-1-3-4(9)5(10)6(11,2-8)12-3/h1-2H2,3-5H,7-11H/t3-,4-,5+,6-/m1/s1
c00096	1.12Beta/C16H25N5O16P2/C17-16-19-12-6(13(28)20-16)18-3-21(12)14-10(26)8(24)5(34-14)2-33-38(29,30)37-39(31,32)36-15-11(27)9(25)7(23)4(1-22)35-15/h1-2H2,3-5H,7-11H,14-15H,22-27H,(H,29,30)(H,31,32)(H3,17,19,20,28)/t4-,5-,7-,8-,9+,10-,11+,14-,15-/m1/s1
c00394	1.12Beta/C16H25N5O16P2/C17-16-19-12-6(13(28)20-16)18-3-21(12)14-10(26)8(24)5(34-14)2-33-38(29,30)37-39(31,32)36-15-11(27)9(25)7(23)4(1-22)35-15/h1-2H2,3-5H,7-11H,14-15H,22-27H,(H,29,30)(H,31,32)(H3,17,19,20,28)/t4-,5-,7-,8-,9+,10-,11-,14?,15?/m1/s1
c01581	1.12Beta/C16H25N5O16P2/C17-16-19-12-6(13(28)20-16)18-3-21(12)14-10(26)8(24)5(34-14)2-33-38(29,30)37-39(31,32)36-15-11(27)9(25)7(23)4(1-22)35-15/h1-2H2,3-5H,7-11H,14-15H,22-27H,(H,29,30)(H,31,32)(H3,17,19,20,28)/t4-,5+,7?,8+,9?,10+,11?,14+,15?/m0/s1
c02280	1.12Beta/C16H25N5O16P2/C17-16-19-12-6(13(28)20-16)18-3-21(12)14-10(26)8(24)5(34-14)2-33-38(29,30)37-39(31,32)36-15-11(27)9(25)7(23)4(1-22)35-15/h1-2H2,3-5H,7-11H,14-15H,22-27H,(H,29,30)(H,31,32)(H3,17,19,20,28)/t4-,5+,7+,8+,9-,10+,11-,14+,15+/m0/s1
c00793	1.12Beta/C3H7N02S/c4-2(1-7)3(5)6/h1H2,2H,4H2,7H,(H,5,6)/t2-/m1/s1
c00736	1.12Beta/C3H7N02S/c4-2(1-7)3(5)6/h1H2,2H,4H2,7H,(H,5,6)
c00097	1.12Beta/C3H7N02S/c4-2(1-7)3(5)6/h1H2,2H,4H2,7H,(H,5,6)/t2-/m0/s1
c00100	1.12Beta/C24H40N7O17P3S/C1-4-15(33)52-8-7-26-14(32)5-6-27-22(36)19(35)24(2,3)10-45-51(42,43)48-50(40,41)44-9-13-18(47-49(37,38)39)17(34)23(46-13)31-12-30-16-20(25)28-11-29-21(16)31/h1-3H3,4-10H2,11-13H,17-19H,23H,34-35H,(H,26,32)(H,27,36)(H,40,41)(H,42,43)(H2,25,28,29)(H2,37,38,39)/t13-,17-,18-,19?,23-/m1/s1

Continued on next page

Table A.6 – continued from previous page

Kegg No.	InChI
c02843	1.12Beta/C24H40N7O17P3S/C1-4-15(33)52-8-7-26-14(32)5-6-27-22(36)19(35)24(2,3)10-45-51(42,43)48-50(40,41)44-9-13-18(47-49(37,38)39)17(34)23(46-13)31-12-30-16-20(25)28-11-29-21(16)31/h1-3H3,4-10H2,11-13H,17-19H,23H,34-35H,(H,26,32)(H,27,36)(H,40,41)(H,42,43)(H2,25,28,29)(H2,37,38,39)/t13-,17-,18-,19?,23-/m1/s1
c02187	1.12Beta/C24H40N7O17P3S/C1-4-15(33)52-8-7-26-14(32)5-6-27-22(36)19(35)24(2,3)10-45-51(42,43)48-50(40,41)44-9-13-18(47-49(37,38)39)17(34)23(46-13)31-12-30-16-20(25)28-11-29-21(16)31/h1-3H3,4-10H2,11-13H,17-19H,23H,34-35H,(H,26,32)(H,27,36)(H,40,41)(H,42,43)(H2,25,28,29)(H2,37,38,39)/t13-,17-,18-,19?,23-/m1/s1



# Appendix B

## MOPAC calculation references

### B.1 Second-protocol inorganic calculations

#### B.1.1 Short atom-atom distances

24 different pairings with short atom-atom distances were found in the 1258 calculations that converged successfully. These are listed below, where each job in which a short distance occurred is provided along with the shortest instance in that job:

**Ba-Ba** ned24-35, 1.73Å; ned24-228, 1.80Å; ned24-359, 1.77Å; ned24-829, 1.77Å;  
ned24-2557, 1.79Å; ned24-2558, 1.69Å; ned24-2611, 1.69Å; ned24-3035, 1.86Å;  
ned24-3195, 1.67Å

**Br-Ta** ned24-3344, 1.88Å

**Ca-Na** ned24-2333, 1.86Å; ned24-3059, 1.86Å

**Ca-S** ned24-2613, 2.03Å

**Cd-N** ned24-2411, 1.51Å

**Cl-Fe** ned24-189, 1.61Å; ned24-3239, 1.04Å; ned24-3240, 1.65Å

**Cl-Hg** ned24-2412, 1.77Å

**Cs-P** ned24-3098, 2.38Å

**F-K** ned24-339, 1.86Å; ned24-339, 1.86Å; ned24-3555, 1.98Å

**Hg-Hg** ned24-523, 2.23Å; ned24-1618, 2.18Å; ned24-2499, 2.20Å; ned24-2739,  
2.21Å

**Hg-N** ned24-2412, 1.68Å

**Hg-Te** ned24-465, 1.97Å; ned24-524, 1.99Å  
**K-Nb** ned24-171, 2.43Å; ned24-339, 2.48Å  
**K-Ta** ned24-340, 2.44Å; ned24-3555, 1.76Å  
**La-La** ned24-801, 1.88Å; ned24-2540, 2.35Å; ned24-2834, 1.97Å  
**Mg-Mg** ned24-2542, 1.88Å  
**Na-Na** ned24-206, 1.22Å; ned24-394, 1.23Å; ned24-413, 1.24Å; ned24-433, 1.26Å;  
 ned24-483, 1.22Å; ned24-774, 1.20Å; ned24-908, 1.22Å; ned24-2394, 1.22Å;  
 ned24-3557, 1.17Å; ned24-3126, 1.19Å; ned24-3128, 1.16Å; ned24-3583, 1.23Å  
**Na-S** ned24-2394, 1.82Å  
**Na-Ta** ned24-3128, 2.13Å  
**P-Pb** ned24-731, 1.88Å  
**Pb-Pb** ned24-2639, 2.10Å  
**Pb-Se** ned24-2639, 1.83Å; ned24-2882, 1.73Å  
**Si-Sr** ned24-828, 1.94Å  
**Zn-Zn** ned24-986, 1.46Å

## B.1.2 Silicas

The 8 phases of silica present in the dataset were:

- low-quartz, ned24-1045
- high-quartz, ned24-1647
- low-tridymite, ned24-1669
- high-tridymite, ned24-2188
- low-cristobalite, ned24-1174
- high-cristobalite, ned24-2461
- coesite, ned24-1056
- stishovite, ned24-1142

### B.1.3 Errors in the data

The different types of data error found, and the corresponding jobs, are:

- missing H atoms - ned24-360, ned24-3387, ned24-3545
- incorrect symmetry elements - ned24-732, ned24-842, ned24-1020, ned24-1021
- incorrect data provided in the CIF - ned24-364, ned24-737, ned24-739

### B.1.4 Errors in modeling

#### Uncontrolled expansion

The job showing uncontrolled expansion is ned24-3016.

#### Large loss of symmetry

The 20 calculations showing a large loss of symmetry are:

ned24-238, ned24-246, ned24-397, ned24-485, ned24-519, ned24-666,  
ned24-1269, ned24-2392, ned24-2515, ned24-2516, ned24-2572, ned24-  
2583, ned24-2584, ned24-2643, ned24-2666, ned24-2693, ned24-2778,  
ned24-2911, ned24-2914, ned24-3354

#### Formation of new bonds

The 4 different atom pairs with more than 1 occurrence of bond formation during calculation are:

**I-I** ned24-353, ned24-397, ned24-404, ned24-2387, ned24-2606, ned24-2667, ned24-  
2678, ned24-2693, ned24-2741, ned24-2839, ned24-2840, ned24-3056, ned24-  
3263, ned24-3290, ned24-3345

**Rb-F** ned24-74, ned24-76, ned24-355

**Al-Si** ned24-541, ned24-1173, ned24-1412

**Tl-O** ned24-469, ned24-3152

## Calculations that terminated with controlled errors

These jobs failed with two different error types. For those elemental crystals discussed in the text, the element symbol is provided next to the job name:

unable to achieve self-consistency ned24-151, ned24-495, ned24-659, ned24-1200, ned14-2541 (Li), ned24-2566 (Li)

numerical problems in bracketing lambda ned24-335, ned24-414, ned24-425, ned24-439, ned24-542, ned24-661, ned24-716 (Au), ned24-791 (Bi), ned24-983, ned24-1890, ned24-2013, ned24-2015, ned24-2016, ned24-2017, ned24-2018, ned24-2021, ned24-2023, ned24-2026, ned24-2029, ned24-2032, ned24-2034, ned24-2100, ned24-2113, ned24-2293, ned24-2529 (Pb), ned24-2531 (Sc), ned24-2533 (Cd), ned24-2549 (Sr), ned24-2551 (Tl), ned24-2552 (Tl), ned24-2577 (Tl), ned24-2597, ned24-2614, ned24-2722, ned24-2803, ned24-2850, ned24-2917 (Bi), ned24-2918 (Bi), ned24-2919 (Bi), ned24-3134, ned24-3417, ned24-3498

## B.2 Organic calculations

### B.2.1 Calculations that terminated with controlled errors

Those calculations that failed with the “numerical problems in bracketing lambda” error and were subsequently found to contain radicals are:

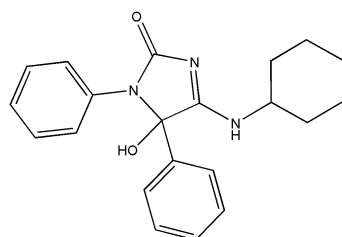
ned24-10476, ned24-11685, ned24-6768, ned24-6770, ned24-6774, ned24-6775, ned24-6846, ned24-6848

Those calculations that ran out of time or reached the cycle limit and were subsequently found to contain radicals are:

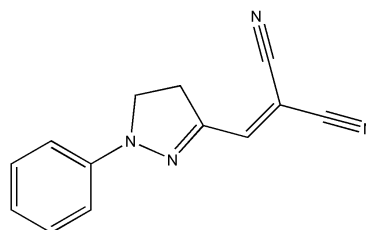
ned24-6773, ned24-6850

The major components for those calculations that failed with the “all convergers are now forced on” error are shown in the table below.

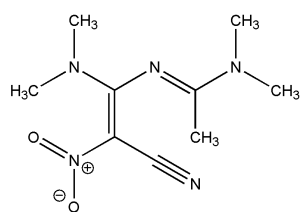
Table B.1: The major components of the structures that had a good starting geometry and caused the “all convergers are now forced on” error.



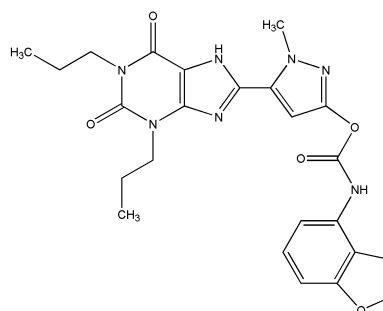
ned24-6589



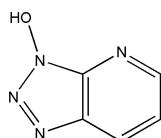
ned24-6956



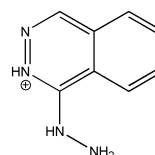
ned24-7236



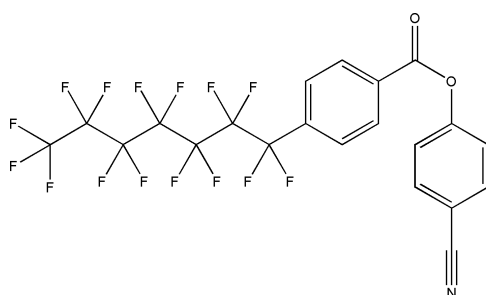
ned24-7413



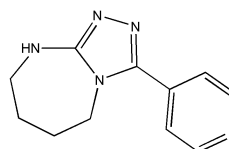
ned24-8575



ned24-8979



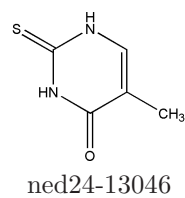
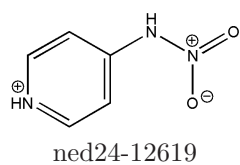
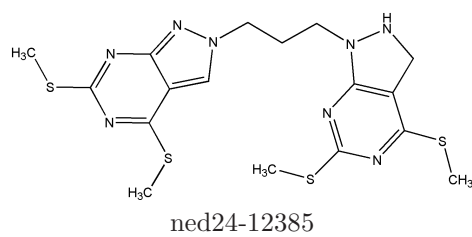
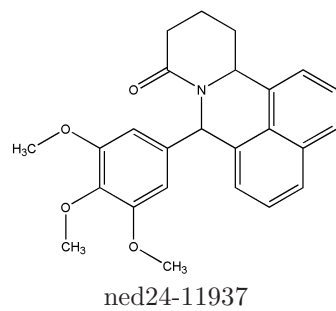
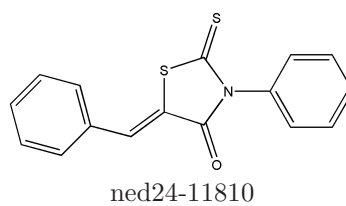
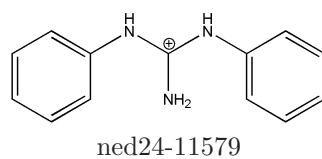
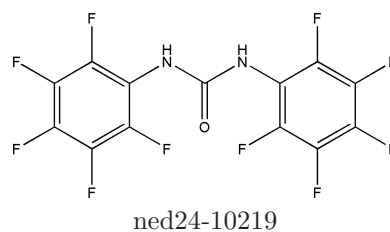
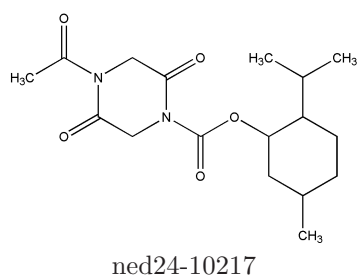
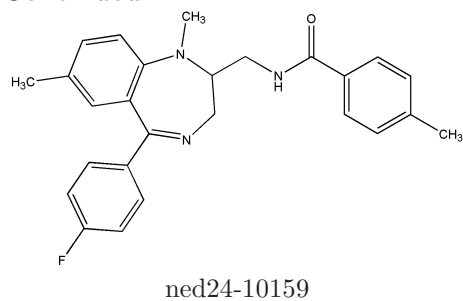
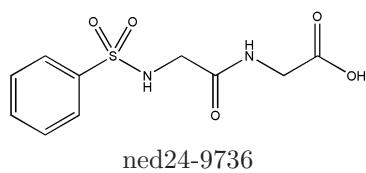
ned24-9055



ned24-9389

Continued on Next Page...

Table B.1 – Continued



## B.2.2 Calculations containing radicals that converged successful

Those calculations that converged successfully, and were subsequently found to contain radicals are:

ned24-6769, ned24-6771, ned24-6843, ned24-6844, ned24-6845, ned24-6849, ned24-6851, ned24-11098

## B.2.3 Changes in connection table

Those structures where a chloride ion involved in a hydrogen bond moves significantly closer to the H are:

ned24-10173; ned24-10577; ned24-11655; ned24-11656; ned24-11963;  
ned24-12008; ned24-12093; ned24-12592; ned24-6602; ned24-9005; ned24-9032; ned24-9566; ned24-9567

The structures for those atom-pairs which are observed to form bonds are:

**S-N** ned24-10052, ned24-12479, ned24-12984

**S-O** ned24-12290, ned24-13165, ned24-6817, ned24-6819

**S-S** ned24-9987, ned24-10034

**S-Cl** ned24-11260, ned24-11585

**Se-Se** ned24-10298, ned24-9782

**Se-I** ned24-8599, ned24-11595, ned24-11665, ned24-11991

**Te-Cl** ned24-12156

**O-Br** ned24-10154, ned24-10319, ned24-11026, ned24-13516

**N-Br** ned24-10683, ned24-10890, ned24-11348, ned24-11594, ned24-12094, ned24-12839, ned24-12854, ned24-12865, ned24-12925, ned24-13060, ned24-7288, ned24-7436

**Br-Br** ned24-11143, ned24-13328

**O-I** ned24-10485, ned24-10499, ned24-12004, ned24-12686, ned24-10720, ned24-12831, ned24-13009, ned24-13127, ned24-13164, ned24-13165, ned24-13354, ned24-13400, ned24-6762, ned24-6772, ned24-7078, ned24-7079, ned24-7080, ned24-7138, ned24-7139, ned24-7153, ned24-7154, ned24-7302, ned24-7316, ned24-7321, ned24-7373, ned24-7374, ned24-7375, ned24-7376, ned24-7416, ned24-8608, ned24-8609, ned24-8610, ned24-8611, ned24-8612, ned24-8929, ned24-9264, ned24-9811, ned24-9883

**S-I** ned24-10826, ned24-11595, ned24-13165

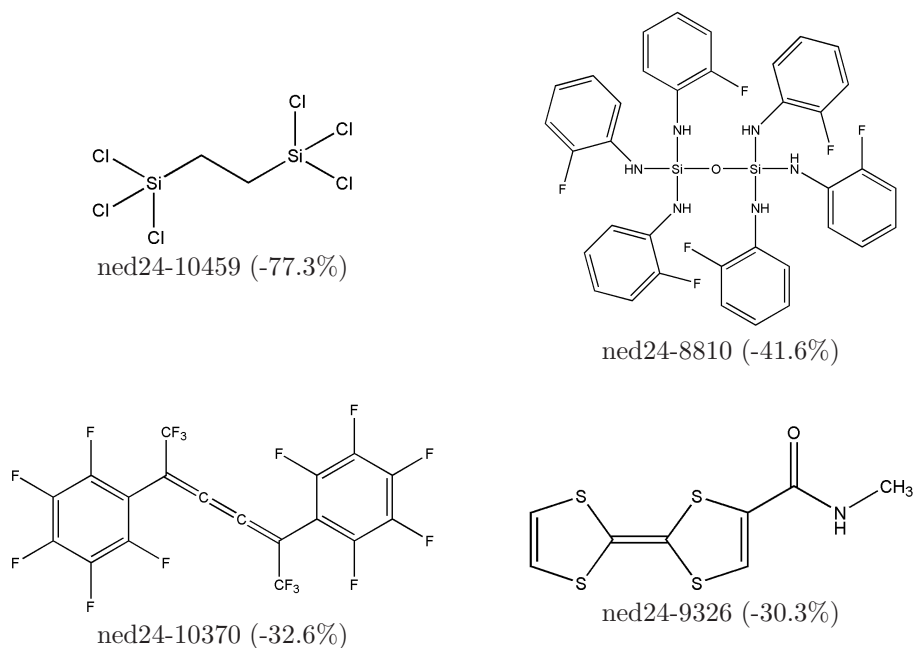
**N-I** ned24-11486, ned24-11670, ned24-11948, ned24-8588

**I-I** ned24-10335, ned24-10485, ned24-11491, ned24-11588, ned24-12984, ned24-13130, ned24-8599

## B.2.4 Density change outliers

The organic structures that were predicted to have density changes of >20% are shown in tables B.2 and B.3. For each structure, the 2D image is provided, as well as the density change predicted by MOPAC and its job number. Each table is organised by predicted density change, starting with the largest.

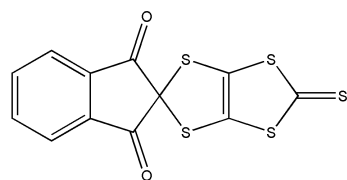
Table B.2: The organic structures which were predicted to have a density change of <-20%



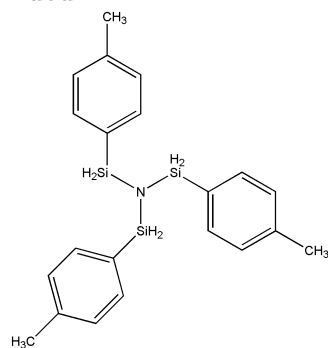
Continued on Next Page...



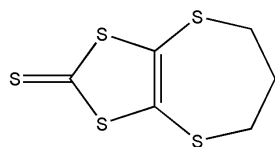
Table B.2 – Continued



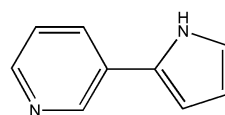
ned24-11230 (-30.1%)



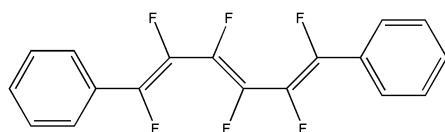
ned24-9514 (-26.1%)



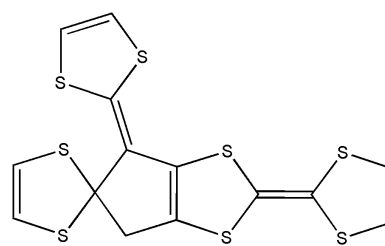
ned24-13335 (-26.0%)



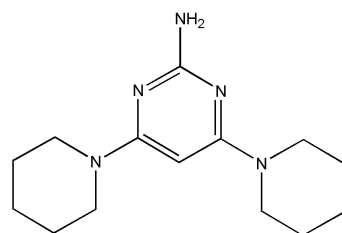
ned24-13200 (-23.0%)



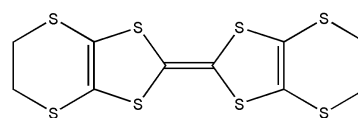
ned24-12186 (-23.0%)



ned24-11144 (-22.7%)



ned24-7257 (-22.3%)

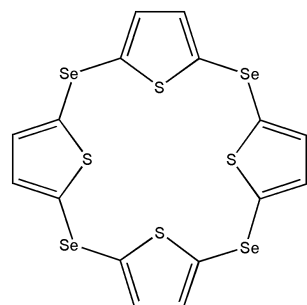


ned24-12062 (-22.0%)

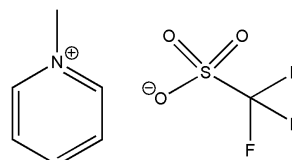
---

Continued on Next Page...

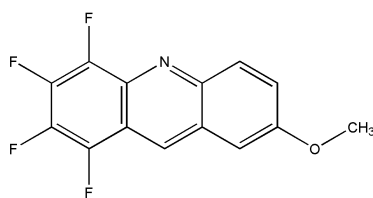
Table B.2 – Continued



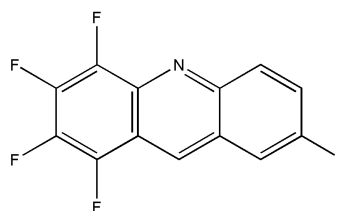
ned24-13067 (-21.8%)



ned24-13177 (-21.1%)

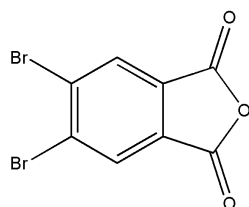


ned24-9160 (-20.9%)

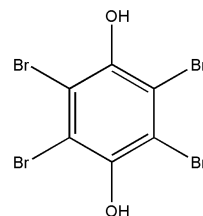


ned24-9163 (-20.1%)

Table B.3: The organic structures which were predicted to have a density change of >20%. Two of the structures, ned24-12762 and ned24-13125, are icosahedral carboranes, and for clarity the 3D image of each has been provided. The key for these two images is: C atoms are grey; H atoms are white, B atoms are pink; O atoms are red and I atoms are purple.



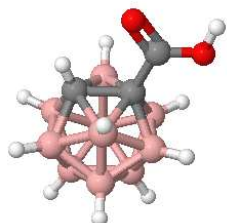
ned24-6801 (28.0%)



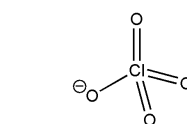
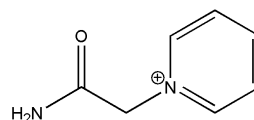
ned24-6827 (26.7%)

Continued on Next Page...

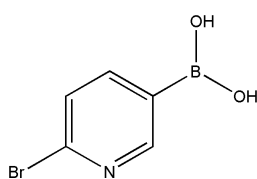
Table B.3 – Continued



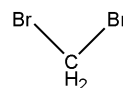
ned24-12762 (24.8%)



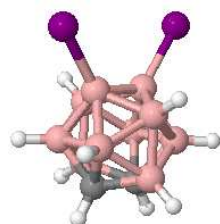
ned24-10214 (24.5%)



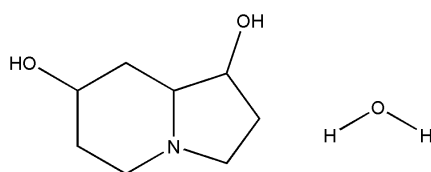
ned24-13146 (24.2%)



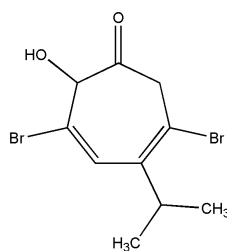
ned24-8644 (24.1%)



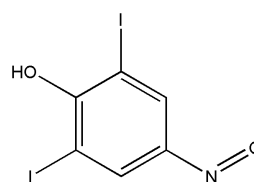
ned24-13125 (23.9%)



ned24-11119 (23.2%)



ned24-13297 (21.1%)

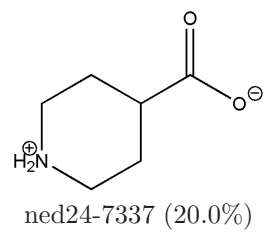
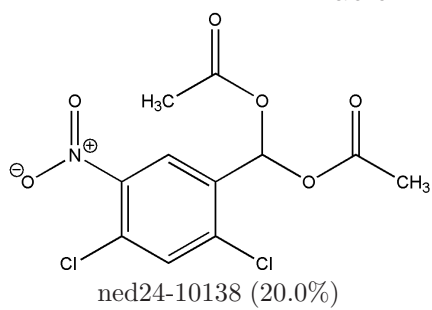


ned24-13020 (20.3%)

---

Continued on Next Page...

Table B.3 – Continued



# Appendix C

## Published Work

The following papers and communications have been published as a result of work contained in this thesis.

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

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

P. T. Corbett, P. Murray-Rust, N. E. Day, J. A. Townsend, H. S. Rzepa, Chemistry publications in CML, *Abstr. Am. Chem. Soc.*, 2006

N. E. Day, P. T. Corbett, P. Murray-Rust, Semantic chemical publishing, *Abstr. Am. Chem. Soc.*, 2007

A. D. Walkingshaw, T. O. H. White, N. E. Day, O. H. Downing, P. Murray-Rust, Representing, indexing and mining scientific data with XML and RDF: Golem and CrystalEye, *Proceedings of XTech 2008*

J. Downing, P. Murray-Rust, A. P. Tonge, P. Morgan, H. S. Rzepa, F. Cotterill, N. Day, Matt J. Harvey, SPECTRa: The Deposition and Validation

of Primary Chemistry Research Data in Digital Repositories, *J. Chem. Inf. Model.*, **ASAP Article**, DOI: 10.1021/ci7004737

# Bibliography

- [1] H. B. Bürgi, J. D. Dunitz, J. M. Lehn, G. Wipff, Stereochemistry of reaction paths at carbonyl centres, *Tetrahedron*, **1974**, 1563-1572
- [2] T. Hey, A. Trefethen, The Data Deluge: An e-Science Perspective, *Wiley*, **2003**, 809–824
- [3] Registry Number and Substance Counts, Chemical Abstracts Service, <http://www.cas.org/cgi-bin/cas/regreport.pl>, accessed on July 17, 2008
- [4] Budapest Open Access Initiative, Open Society Institute and Soros Foundation Network, <http://www.soros.org/openaccess/>, accessed on July 2, 2008
- [5] Bethesda Statement on Open Access Publishing, The SPARC Open Access Newsletter, <http://www.earlham.edu/~peters/fos/bethesda.htm>, accessed on July 2, 2008
- [6] Berlin Declaration on Open Access to Knowledge in the Sciences and Humanities, Conference on Open Access to Knowledge in the Sciences and Humanities, <http://oa.mpg.de/openaccess-berlin/berlindeclaration.html>, accessed on July 2, 2008
- [7] Homepage, DSpace, <http://www.dspace.org/>, accessed on July 2, 2008
- [8] Open Access and Institutional Repositories with EPrints, EPrints, <http://www.eprints.org/>, accessed on July 4, 2008

- [9] Open Data, Wikipedia, [http://en.wikipedia.org/wiki/Open\\_data](http://en.wikipedia.org/wiki/Open_data), accessed on July 4, 2008
- [10] Homepage, Science Commons, <http://sciencecommons.org/>, accessed on July 4, 2008
- [11] Homepage, Open Data Commons, <http://www.opendatacommons.org/>, accessed on July 4, 2008
- [12] Homepage, Nucleic Acids Research, <http://nar.oxfordjournals.org/>, accessed on July 9, 2008
- [13] Homepage, Acta Crystallographica Section E, Structure Reports, <http://journals.iucr.org/e/journalhomepage.html>, accessed on July 9, 2008
- [14] Open Access resolution adopted by the Harvard Faculty of Arts and Sciences, Faculty of Arts and Sciences, Harvard University, [http://www.fas.harvard.edu/~secfas/May\\_2008\\_Agenda.pdf](http://www.fas.harvard.edu/~secfas/May_2008_Agenda.pdf), accessed on July 9, 2008
- [15] Homepage, PubChem, <http://pubchem.ncbi.nlm.nih.gov/>, accessed on July 9, 2008
- [16] Homepage, Wikipedia, <http://en.wikipedia.org/>, accessed on July 13, 2008
- [17] Homepage, Wichempedia, <http://www.wichempedia.org/Search.aspx?q=all>, accessed on July 13, 2008
- [18] D. L. Wheeler, T. Barrett, D. A. Benson, S. H. Bryant, K. Canese, V. Chetvernin, D. M. Church, M. DiCuccio, R. Edgar, S. Federhen, L. Y. Geer, Y. Kapustin, O. Khovayko, D. Landsman, D. J. Lipman, T. L. Madden, D. R. Maglott, J. Ostell, V. Miller, K. D. Pruitt, G. D. Schuler, E. Sequeira, S. T. Sherry, K. Sirotkin, A. Souvorov, G. Starchenko, R. L. Tatusov, T. A. Tatusova, L. Wagner, E. Yaschenko, Database resources of the National Center for Biotechnology Information, *Nucleic Acids Research*, **35**, D5-D12



- [19] J. L. Sussman, D. Lin, J. Jiang, N. O. Manning, J. Prilusky, O. Ritter, E. E. Abola, Protein Data Bank (PDB): Database of Three-Dimensional Structural information of Biological Macromolecules, *Acta Cryst.*, **D54**, 1078-1084
- [20] A. Hamosh, A. F. Scott, J. Amberger, C. Bocchini, D. Valle, V. A. McKusick, Online Mendelian Inheritance in Man (OMIM): a knowledgebase of human genes and genetic disorders, *Nucleic Acids Research*, **30**, 52-55
- [21] S. R. Hall, F. H. Allen, I. D. Brown, The crystallographic information file (CIF): a new standard archive file for crystallography, *Acta. Cryst.*, **A47**, 655–685
- [22] Extracting the science from science publications, University of Cambridge Computer Laboratory, <http://www.cl.cam.ac.uk/~aac10/escience/sciborg.html>, accessed on July 9, 2008
- [23] P. Murray-Rust, H. S. Rzepa, The Next Big Thing: From Hypermedia to Datuments, *J. Digital. Inf.*, **2004**, 248
- [24] RSC Prospect, RSC Publishing, <http://www.rsc.org/Publishing/Journals/ProjectProspect/>, accessed on July 9, 2008
- [25] T. Berners-Lee, J. Hendler, O. Lassila, The Semantic Web, *Sci. Am.*, **2001**, 284, 34-43
- [26] Semantic Web: Linked Data on the Web, World Wide Web Consortium Talks, [http://www.w3.org/2007/Talks/0130-sb-W3CTechSemWeb/#\(24\)](http://www.w3.org/2007/Talks/0130-sb-W3CTechSemWeb/#(24)), accessed on July 9, 2008
- [27] 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
- [28] P. Murray-Rust and H. S. Rzepa, Chemical Markup, XML and the World-Wide Web. 4. CML Schema, *J. Chem. Inf. Comput. Sci.*, **2003**, 43, 757-772

- [29] S. Kuhn, T. Helmus, R. J. Lancashire, P. Murray-Rust, H. S. Rzepa, C. Steinbeck, E. L. Willighagen, Chemical Markup, XML, and the World Wide Web. 7. CMLSpect, an XML Vocabulary for Spectral Data, *J. Chem. Inf. Model.*, **47**, 2015-2034
- [30] Homepage, XOM, <http://www.xom.nu/>
- [31] Defining e-Science, National e-Science Centre, <http://www.nesc.ac.uk/nesc/define.html>, accessed on July 6, 2008
- [32] Homepage, Folding@home project, <http://folding.stanford.edu/>, accessed on July 6, 2008
- [33] Client statistics by OS, Folding@home project, <http://fah-web.stanford.edu/cgi-bin/main.py?ctype=osstats>, accessed on July 6, 2008
- [34] IBM Triples Performance of World's Fastest, Most Energy-Efficient Supercomputer, IBM Press room, <http://www-03.ibm.com/press/us/en/pressrelease/21791.wss>, accessed on July 6, 2008
- [35] Homepage, National e-Science Centre, <http://www.nesc.ac.uk/>, accessed on July 6, 2008
- [36] Homepage, Cambridge e-Science Centre, <http://www.escience.cam.ac.uk/>, accessed on July 6, 2008
- [37] Homepage, CamGrid project, <http://www.escience.cam.ac.uk/projects/camgrid/>, accessed on July 6, 2008
- [38] The Open Source Definition, The Open Source Initiative, <http://opensource.org/docs/osd>, accessed on July 15, 2008
- [39] Homepage, The Chemistry Development Kit (CDK), [http://almost.cubic.uni-koeln.de/cdk/cdk\\_top](http://almost.cubic.uni-koeln.de/cdk/cdk_top), accessed on July 15, 2008
- [40] Homepage, Open Babel: The Open Source Chemistry Toolbox, [http://openbabel.org/wiki/Main\\_Page](http://openbabel.org/wiki/Main_Page), accessed on July 15, 2008

- [41] Homepage, Jmol: an open-source Java viewer for chemical structures in 3D, <http://jmol.sourceforge.net/>, accessed on July 15, 2008
- [42] Experiment on search engine spider behaviour involving more than 2 billion webpages, Drunk Men Work Here, <http://drunkmenworkhere.org/219>, accessed on April 11, 2008
- [43] Homepage, Sitemaps, <http://www.sitemaps.org/>, accessed on April 11, 2008
- [44] Methanol — Compound Summary (CID: 887), PubChem, <http://pubchem.ncbi.nlm.nih.gov/summary/summary.cgi?cid=887>, accessed on April 11, 2008
- [45] CAS REGISTRY and CAS Registry Numbers, Chemical Abstracts Service, <http://www.cas.org/expertise/cascontent/registry/regsys.html>, accessed on April 11, 2008
- [46] Homepage, Registry of Toxic Effects of Chemical Substances (RTECS), <http://www.cdc.gov/niosh/rtecs/>, accessed on April 11, 2008
- [47] D. Weininger, SMILES, a Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules, *J. Chem. Inf. Comp. Sci.*, **1988**, 28, 31–36
- [48] D. Weininger, A. Weininger, J. L. Weininger, SMILES. 2. Algorithm for Generation of Unique SMILES Notation, *J. Chem. Inf. Comp. Sci.*, **1989**, 29, 97–101
- [49] Unofficial InChI FAQ, World Wide Molecular Matrix, <http://wwmm.ch.cam.ac.uk/inchifaq>, accessed on April 11, 2008
- [50] S. E. Stein, S. R. Heller and D. Tchekovskoi, An Open Standard for Chemical Structure Representation: The IUPAC Chemical Identifier, *proceedings of the International Chemical Information Conference (Nimes)*, **2003**

- [51] Project: IUPAC - International Chemical Identifier, IUPAC, <http://www.iupac.org/projects/2000/2000-025-1-800.html>, accessed on April 11, 2008
- [52] eCrystals, University of Southampton Chemistry Department, <http://ecrystals.chem.soton.ac.uk/>, accessed on April 11, 2008
- [53] InChI 1.02beta Software Release — introducing InChIKey, IUPAC, <http://www.iupac.org/inchi/release102.html>, accessed on April 11, 2008
- [54] S. Heller, The future of linking (most all) literature and scientific chemical information using InChI and InChIKey, *Wiley-Weinheim seminar*, **2007**
- [55] Homepage, chem-bla-ics weblog, <http://chem-bla-ics.blogspot.com/>, accessed on April 11, 2008
- [56] Homepage, Totally Synthetic weblog, <http://totallysynthetic.com/blog/>, accessed on April 11, 2008
- [57] Homepage, Useful Chemistry weblog, <http://usefulchem.blogspot.com/>, accessed on April 11, 2008
- [58] Homepage, ChemAxon MarvinSketch, <http://www.chemaxon.com/product/msketch.html>, accessed on April 11, 2008
- [59] Google SOAP Search API (Beta), Google Code, <http://code.google.com/apis/soapsearch/reference.html>, accessed on April 11, 2008
- [60] Google SOAP Search API (Beta) Search Results Format, Google Code, [http://code.google.com/apis/soapsearch/reference.html#search\\_results\\_format](http://code.google.com/apis/soapsearch/reference.html#search_results_format), accessed on April 11, 2008
- [61] InChImatic homepage, <http://inchimatic.com/>, accessed on April 11, 2008

- [62] S.Omura, Y.Iwai, A.Hirano, A.Nakagawa, J.Awaya, H.Tsuchiya, Y.Takahashi and R.Masuma, *J. Antibiotics*, **1977**, 30, 275
- [63] A.Furasaki, N.Hashiba, T.Matsumoto, A.Hirano, Y.Iwai, and S.Omura, *Chem. Comm.*, **1978**, 801
- [64] N.Funato, H.Takayanagi, Y.Kouda, Y.Toda, H.Harigaya, Y.Iwai and S.Omura, *Tetrahedron Letters*, **1994**, 35, 1251
- [65] Staurosporine, Chemsoc, <http://www.chemsoc.org/exemplarchem/entries/jagfin/jagfin/contents/therap.htm>, accessed on April 11, 2008
- [66] Staurosporine, LC Laboratories, <http://www.lclabs.com/PRODFILE/S-Z/S-9300.php4>, accessed on April 11, 2008
- [67] Staurosporine, National Institute of Allergies and Diseases, [http://apps1.niaid.nih.gov/struct\\_search/images/structures/001539.gif](http://apps1.niaid.nih.gov/struct_search/images/structures/001539.gif), accessed on April 11, 2008
- [68] Staurosporine, European Bioinformatics Institute, [http://www.ebi.ac.uk/msd-srv/chempdb/cgi-bin/cgi.pl?FUNCTION=record&ENTITY=CHEM\\_COMP&PRIMARYKEY=2989&PARENTINDEX=-1&APPLICATION=1](http://www.ebi.ac.uk/msd-srv/chempdb/cgi-bin/cgi.pl?FUNCTION=record&ENTITY=CHEM_COMP&PRIMARYKEY=2989&PARENTINDEX=-1&APPLICATION=1), accessed on April 11, 2008
- [69] Staurosporine, Volkerberl, [http://www.volkerberl.de/content/english/2\\_research/navi\\_inhalt/inhalt\\_research\\_postdoc\\_background.htm](http://www.volkerberl.de/content/english/2_research/navi_inhalt/inhalt_research_postdoc_background.htm), accessed on April 11, 2008
- [70] Staurosporine, Cope with Cytokines, <http://www.copewithcytokines.de/cope.cgi?key=Staurosporine>, accessed on April 11, 2008
- [71] Staurosporine — Compound Summary (CID: 5279), PubChem, <http://pubchem.ncbi.nlm.nih.gov/summary/summary.cgi?cid=5279>, accessed on April 11, 2008
- [72] Staurosporine, Turku University of Applied Sciences, <http://www.turkuamk.fi/varsta/lividans.htm>, accessed on April 11, 2008

- [73] Staurosporine — Compound Summary (CID: 358479), PubChem, <http://pubchem.ncbi.nlm.nih.gov/summary/summary.cgi?cid=358479>, accessed on April 11, 2008
- [74] Staurosporine, Upstate, <http://www.upstate.com/img/coa/19-123-29999.pdf>, accessed on April 11, 2008
- [75] Staurosporine, CaymanChem, <http://www.caymanchem.com/app/template/Product.vm/catalog/81590>, accessed on April 11, 2008
- [76] Staurosporine, EMD Biosciences, <http://www.emdbiosciences.com/product/569397>, accessed on April 11, 2008
- [77] Staurosporine, Instituto Biomar, <http://www.institutobiomar.com/stock.htm>, accessed on April 11, 2008
- [78] Staurosporine, National Institute of Allergies and Diseases, [http://apps1.niaid.nih.gov/struct\\_search/class/class\\_many.asp?class=PROTEIN%20KINASE%20C%20INHIBITORS](http://apps1.niaid.nih.gov/struct_search/class/class_many.asp?class=PROTEIN%20KINASE%20C%20INHIBITORS), accessed on April 11, 2008
- [79] Protein Kinase Inhibitors : Staurosporine, proteinkinase, [http://www.proteinkinase.de/html/protein\\_kinase\\_inhibitors.html#staurosporine](http://www.proteinkinase.de/html/protein_kinase_inhibitors.html#staurosporine), accessed on April 11, 2008
- [80] Staurosporine from Streptomyces sp. (S5921), Sigma Aldrich, <http://www.sigmaaldrich.com/catalog/search/ProductDetail/SIGMA/S5921>, accessed on April 11, 2008
- [81] Staurosporine from Streptomyces sp. (85658), Sigma Aldrich, <http://www.sigmaaldrich.com/catalog/search/ProductDetail/FLUKA/85658>, accessed on April 11, 2008
- [82] Staurosporine from Streptomyces sp. (S4400), Sigma Aldrich, <http://www.sigmaaldrich.com/catalog/search/ProductDetail/SIGMA/S4400>, accessed on April 11, 2008

- [83] Staurosporine, Visiscience, <http://visiscience.com/%20samples/Staurosporine.jpg>, accessed on April 11, 2008
- [84] Staurosporine, Serva Electrophoresis, [http://www.serva.de/servaWeb/www\\_root/ar03/templates/Ar03ProductDetail.jsp?artNr=35385](http://www.serva.de/servaWeb/www_root/ar03/templates/Ar03ProductDetail.jsp?artNr=35385), accessed on April 11, 2008
- [85] Staurosporine, Alomone Labs, [http://www.alomone.com/p\\_postcards/database/104.htm](http://www.alomone.com/p_postcards/database/104.htm), accessed on April 11, 2008
- [86] Staurosporine, ChemExper, <http://www.chemexper.com/chemicals/supplier/cas/62996-74-1.html>, accessed on April 11, 2008
- [87] Staurosporine — Compound Summary (CID: 44259), PubChem, <http://pubchem.ncbi.nlm.nih.gov/summary/summary.cgi?cid=44259>, accessed on April 11, 2008
- [88] Staurosporine — Compound Summary (CID: 44259), PubChem, <http://pubchem.ncbi.nlm.nih.gov/summary/summary.cgi?cid=451705>, accessed on April 11, 2008
- [89] Homepage, ChemSpider, <http://www.chemspider.com/>, accessed on April 11, 2008
- [90] Lecture notes from Data Exchange, Quality Assurance and Integrated Data Publication (CIF and checkCIF), IUCr submission for the ALPSP Award for Publishing Innovation, 2006, [http://www.iucr.org/iucr-top/docs/presentations/ALPSP\\_innovation2006.ppt](http://www.iucr.org/iucr-top/docs/presentations/ALPSP_innovation2006.ppt), accessed on June 30, 2008
- [91] IUCr Crystallographic Information Framework, IUCr, <http://www.iucr.org/iucr-top/cif/index.html>, accessed on May 3, 2008
- [92] S. R. Hall, The STAR File: A New Format for Electronic Data Transfer and Archiving, *J. Chem. Inf. Comput. Sci.*, **31**, 326-333

- [93] I. D. Brown, CIF (Crystallographic Information File): A Standard for Crystallographic Data Interchange, *J. Res. Natl. Inst. Stand. Technol.*, **101**, 341
- [94] CIF Core dictionary (coreCIF) — version 2.3.2, IUCr, [http://www.iucr.org/iucr-top/cif/cifdic\\_html/1/cif\\_core.dic/index.html](http://www.iucr.org/iucr-top/cif/cifdic_html/1/cif_core.dic/index.html), accessed on May 3, 2008
- [95] CIF Macromolecular dictionary (mmCIF) — version 2.0.09, IUCr, [http://www.iucr.org/iucr-top/cif/cifdic\\_html/2/cif\\_mm.dic/index.html](http://www.iucr.org/iucr-top/cif/cifdic_html/2/cif_mm.dic/index.html), accessed on May 3, 2008
- [96] S. R. Hall, A. P. F. Cook, STAR Dictionary Definition Language: Initial Specification, *J. Chem. Inf. Comput. Sci.*, **35**, 819-825
- [97] Dictionary Definition Language 1 homepage, IUCr, <http://www.iucr.org/iucr-top/cif/ddl1/index.html>, accessed on May 3, 2008
- [98] Dictionary Definition Language 2 homepage, IUCr, <http://www.iucr.org/iucr-top/cif/ddl2/index.html>, accessed on May 5, 2008
- [99] I. D. Brown, B. McMahon, CIF: The Computer Language of Crystallography, *Acta. Cryst.*, **B58**, 317-324
- [100] checkCIF, IUCr, <http://checkcif.iucr.org/>, accessed on May 5, 2008
- [101] Details of checkCIF/PLATON tests, IUCr, <http://journals.iucr.org/services/cif/datavalidation.html>, accessed on May 3, 2008
- [102] S. R. Hall, H. J. Bernstein, CIF Applications. V. CIFtbx2: extended tool box for manipulation CIFs, *J. Appl. Cryst.*, **1996**, 29, 598-603
- [103] J. D. Westbrook, S.-H. Hsieh, P. M. D. Fitzgerald, CIF Applications. VI. CIFLIB: an application program interface to CIF dictionaries and data files, *J. Appl. Cryst.*, **1997**, 30, 79-83
- [104] H. J. Bernstein, F. C. Bernstein, P. E. Bourne, CIF Applications. VIII. pdb2cif: translating PDB entries into mmCIF format, *J. Appl. Cryst.*, **1998**, 31, 278-281



- [105] P. R. Edgington, HICCuP: High-Integrity CIF Checking Using Python, *Cambridge Crystallographic Data Centre, UK*, **1997**
- [106] J. R. Hester, A validating CIF parser: PyCIFRW, *J. Appl. Cryst.*, **2006**, 39, 621–625
- [107] W. Bluhm, Star (CIF) parser, San Diego Supercomputer Centre, <http://pdb.sdsc.edu/STAR/index.html>, accessed on May 3, 2008
- [108] About SAX, SAX project, <http://www.saxproject.org/>, accessed on May 3, 2008
- [109] Document Object Model (DOM), W3C, <http://www.w3.org/DOM/>, accessed on May 3, 2008
- [110] Java, Sun Developer Network, <http://www.javasoft.com>, accessed on May 4, 2008
- [111] Chemical Markup Language, Sourceforge, <http://www.sf.net/projects/cml>, accessed on May 3, 2008
- [112] Chemical Markup Language Subversion, Sourceforge, [http://sourceforge.net/svn/?group\\_id=51361](http://sourceforge.net/svn/?group_id=51361), accessed on May 3, 2008
- [113] Welcome to Maven, Apache Maven Project, <http://maven.apache.org/>, accessed on May 3, 2008
- [114] legacy2cml download page, Sourceforge, [http://sourceforge.net/project/showfiles.php?group\\_id=51361&package\\_id=232633](http://sourceforge.net/project/showfiles.php?group_id=51361&package_id=232633), accessed on May 3, 2008
- [115] Introduction, Jakarta Commons HTTPClient, <http://hc.apache.org/httpclient-3.x/>, accessed on May 3, 2008
- [116] Tagsoup, Chester County InterLink, <http://home.ccil.org/~cowan/XML/tagsoup/>, accessed on May 3, 2008

- [117] CIF - Common Semantic Features, IUCr, <http://www.iucr.org/iucr-top/cif/spec/version1.1/cifsemantics.html>, accessed on May 3, 2008
- [118] Homepage, Blue Obelisk, [http://blueobelisk.sourceforge.net/wiki/Main\\_Page](http://blueobelisk.sourceforge.net/wiki/Main_Page), accessed on May 3, 2008
- [119] The Blue Obelisk Data Repository download page, Sourceforge, [http://sourceforge.net/project/showfiles.php?group\\_id=189199](http://sourceforge.net/project/showfiles.php?group_id=189199), accessed on May 3, 2008
- [120] `_atom_site_disorder_assembly` definition in the coreCIF dictionary, IUCr, [http://www.iucr.org/iucr-top/cif/cifdic\\_html/1/cif\\_core\\_dic/Iatom\\_site\\_disorder\\_assembly.html](http://www.iucr.org/iucr-top/cif/cifdic_html/1/cif_core_dic/Iatom_site_disorder_assembly.html), accessed on May 3, 2008
- [121] J. S. Rollett, Computing Methods in Crystallography, Oxford, *Pergamon Press*, **1965**, p23
- [122] `_atom_site_label` definition in the Core CIF Dictionary (version 2.3), IUCr, [http://www.iucr.org/iucr-top/cif/cif\\_core/definitions/Cdata\\_atom\\_site\\_label.html](http://www.iucr.org/iucr-top/cif/cif_core/definitions/Cdata_atom_site_label.html), accessed on May 3, 2008
- [123] R. Köppen, F. Emmerling, R. Becker, Decabromodiphenylethane, *Acta Cryst.*, **E63**, o585-586
- [124] L. Xu, W. -Z. Bi, K. Zhou, (*E*)-Ethyl 2-[2-(bromomethyl)phenyl]-2-(methoxyimino)acetate, *Acta. Cryst.*, **E63**, o4686
- [125] V. Pavlyuk, P. Solokha, G. Dmytriv, B. Marciniak, V. Paul-Boncour, The Heusler-type alloy MgZn<sub>2</sub>Ce, *Acta. Cryst.*, **E63**, i161
- [126] J. Miklovic, J. Moncol, D. Miklos, P. Segla, M. Koman, *trans*-Tetraaquabis[3-(3-pyridyl)acrylato- $\kappa N$ ]cobalt(II), *Acta. Cryst.*, **E64**, m426
- [127] W. -C. Song, M. -J. Zhang, Y. Tao, J. -R. Li, *catena*-Poly[[bis(pyridine- $\kappa N$ )nickel(II)]-di- $\mu$ -azido- $\kappa^4 N^1:N^3$ ]-[bis(pyridine- $\kappa N$ )nickel(II)]-di- $\mu$ -azido- $\kappa^4 N^1:N^1$ ], *Acta. Cryst.*, **E63**, m3062

- [128] jni-InChI, Sourceforge, <http://jni-inchi.sourceforge.net/>, accessed on May 3, 2008
- [129] CrystalEye, World Wide Molecular Matrix, <http://wwmm.ch.cam.ac.uk/crystaleye/index.html>, accessed on May 3, 2008
- [130] J. Downing, P. Murray-Rust, A. P. Tonge, P. Morgan, H. S. Rzepa, F. Cotterill, N. Day, Matt J. Harvey, SPECTRa: The Deposition and Validation of Primary Chemistry Research Data in Digital Repositories, *J. Chem. Inf. Model.*, **ASAP Article**, DOI: 10.1021/ci7004737
- [131] Homepage, Cambridge Crystallographic Data Centre, <http://www.ccdc.cam.ac.uk/index.php>, accessed on May 14, 2008
- [132] F. H. Allen, The Cambridge Structural Database: a quarter of a million crystal structures and rising, *Acta. Cryst.*, **B58**, 380–388
- [133] Cambridge Structural Database, Cambridge Crystallographic Data Centre, <http://www.ccdc.cam.ac.uk/products/csd/>, accessed on May 14, 2008
- [134] T. Oinn, M. Greenwood, M. Addis, M. N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. R. Pocock, M. Senger, R. Stevens, A. Wipat, C. Wroe, Taverna: Lessons in creating a workflow environment for the life sciences, *Grid Workflow Special Issue*, **2005**, 1067-1100
- [135] L. Wang, J. J. Riethoven, A. Robinson, XEMBL: distributing EMBL data in XML format, *Bioinformatics*, **18**, 1147–1148
- [136] Bibliographic Query Service, European Bioinformatics Institute, <http://industry.ebi.ac.uk/openBQS/>, accessed on May 14, 2008
- [137] M. Senger, P. Rice, T. Oinn, SoapLab: a unified Sesame door to analysis tools, *Proceedings of the UK e-Science All Hands Meeting*, **2003**
- [138] Beanshell homepage, <http://www.beanshell.org/>

- [139] WWMM Web Services, WWMM, <http://wwmm-svc.ch.cam.ac.uk/wwmm/html/>, accessed on May 14, 2008
- [140] WWMM OpenBabel Web Service, WWMM, <http://wwmm-svc.ch.cam.ac.uk/wwmm/html/observer.html>, accessed on May 14, 2008
- [141] WWMM InChI Web Service, WWMM, <http://wwmm-svc.ch.cam.ac.uk/wwmm/html/inchiserver.html>, accessed on May 14, 2008
- [142] M. Pierce, oral presentation, Community Grids Lab Work, *CICC Quarterly Meeting*, **2006**, [http://grids.ucs.indiana.edu/ptliupages/presentations/CICCMarlon\\_Jan27\\_06.ppt](http://grids.ucs.indiana.edu/ptliupages/presentations/CICCMarlon_Jan27_06.ppt), accessed on May 14, 2008
- [143] J. Kim, oral presentation, Building Services for BCI with Taverna, *CICC Quarterly Meeting*, **2006**, [http://grids.ucs.indiana.edu/ptliupages/presentations/CICCJake\\_Jan27\\_06.ppt](http://grids.ucs.indiana.edu/ptliupages/presentations/CICCJake_Jan27_06.ppt), accessed on May 14, 2008
- [144] Freefluo, Sourceforge, <http://freefluo.sourceforge.net/>, accessed on May 14, 2008
- [145] Homepage, Eclipse - an open development platform, <http://www.eclipse.org/>, accessed on May 14, 2008
- [146] R. T. Fielding, Architectural Styles and the Design of Network-based Software Architectures, *PhD Thesis* **2000**
- [147] Web Scraping, Wikipedia, [http://en.wikipedia.org/wiki/Web\\_scraping](http://en.wikipedia.org/wiki/Web_scraping), accessed on May 14, 2008
- [148] Crystallography Open Database homepage, <http://cod.ibtl.it/>, accessed on May 14, 2008
- [149] Database ZIP file, Crystallography Open Database, <http://www.crystallography.net/cod/cod.zip>, accessed on May 14, 2008
- [150] HTTP Request fields — Hypertext Transfer Protocol Version 1.x, W3C, <http://www.w3.org/Protocols/HTTP/HTRQ-Headers.html>, accessed on May 14, 2008

- [151] D. -X. Zhu, W. Sun, G. -F. Yang, S. W. Ng, *rac*-Bis[1-(9,9-dioxo-10*H*-phenothiazin-10-yl)-2-propyl]dimethylammonium terephthalate trihydrate, *Acta. Cryst.*, **E63**, o4830
- [152] CrystalEye, World Wide Molecular Matrix <http://wwmm.ch.cam.ac.uk/crystaleye/>, accessed on May 14, 2008
- [153] Greasemonkey, Firefox Add-ons, <https://addons.mozilla.org/en-US/firefox/addon/748>, accessed on May 14, 2008
- [154] Homepage, Mozilla Firefox, <http://www.mozilla.com/en-US/firefox/>, accessed on May 14, 2008
- [155] The XMLHttpRequest Object Working Draft, W3C, <http://www.w3.org/TR/XMLHttpRequest/>, accessed on May 14, 2008
- [156] E. L. Willighagen, N. M. O’Boyle, H. Gopalakrishnan, D. Jiao, R. Guha, C. Steinbeck, D. J. Wild, Userscripts for the Life Sciences, *BMC Bioinformatics*, **8**:487
- [157] CrystalEye Greasemonkey, Userscripts, <http://userscripts.org/scripts/show/11439>, accessed on May 14, 2008
- [158] RSS 2.0 feed for structures containing Carbon-Silicon bonds, CrystalEye, [http://wwmm.ch.cam.ac.uk/crystaleye/feed/bonds/C-Si/rss/rss\\_20/feed.xml](http://wwmm.ch.cam.ac.uk/crystaleye/feed/bonds/C-Si/rss/rss_20/feed.xml), accessed on May 14, 2008
- [159] Homepage, Google Reader, <http://reader.google.com>, accessed on May 14, 2008
- [160] Homepage, Bioclipse, <http://www.bioclipse.net/>, accessed on May 14, 2008
- [161] P. Murray-Rust, H. S. Rzepa, M. J. Williamson, E. L. Willighagen, Chemical Markup, XML and the World Wide Web. Part 5. Applications of Chemical Metadata in RSS Aggregators, *J. Chem. Inf. Comp. Sci.*, **44**, 462-469

- [162] Homepage, StAX (The Streaming API for XML), <http://stax.codehaus.org/>, accessed on May 14, 2008
- [163] Atom Syndication Format specification, AtomEnabled, <http://atomenabled.org/developers/syndication/atom-format-spec.php>, accessed on May 14, 2008
- [164] Homepage, del.icio.us, <http://del.icio.us/>, accessed on May 14, 2008
- [165] J. Townsend, Automated Analysis and Validation of Chemical Literature, *PhD Thesis* **2007**
- [166] Atom feed paging and archiving, IETF Tools, <http://tools.ietf.org/html/rfc5005>, accessed on May 14, 2008
- [167] Atom Archived Feeds, IETF Tools, <http://tools.ietf.org/html/rfc5005#section-4>, accessed on May 14, 2008
- [168] Atom History vs Large Archive Files, Coding Trombonist, <http://wwmm.ch.cam.ac.uk/blogs/downing/?p=140>, accessed on May 14, 2008
- [169] Homepage, The Golem ontology system, <http://www.lexical.org.uk/science/golem/>, accessed on May 14, 2008
- [170] P. T. Corbett, P. Murray-Rust, Using chemical structure in open-source chemical text mining, *Chemistry Central Journal*, **2008**, **2**(Suppl 1):S10
- [171] Profile For Using the Keyword "meta" As an XHTML/HTML Link Type, inamidst, <http://inamidst.com/misc/uriprofile>, accessed on May 14, 2008
- [172] RDFa primer, W3C, <http://www.w3.org/TR/xhtml1-rdfa-primer/>, accessed on May 14, 2008
- [173] The geographic spread of crystallography, Brighten the Corners, <http://wwmm.ch.cam.ac.uk/blogs/walkingshaw/?p=53>, accessed on May 14, 2008

- [174] Crowdsourcing, Wikipedia, <http://en.wikipedia.org/wiki/Crowdsourcing>, accessed on May 14, 2008
- [175] Antony Williams, ChemSpider Has Curated Over 500 Comments, ChemSpider Blog, <http://www.chemspider.com/blog/chemspider-has-curated-over-500-comments.html>, accessed on May 14, 2008
- [176] Using Connotea for CrystalEye data checking, Coding Trombonist, <http://wmm.ch.cam.ac.uk/blogs/downing/?p=171>, accessed on May 14, 2008
- [177] Homepage, Connotea, <http://www.connotea.org/>, accessed on May 14, 2008
- [178] J. J. P. Stewart, Optimization of parameters for semiempirical methods V: Modification of NDDO approximations and application to 70 elements, *J. Mol. Model.*, **13**, 1173-1213
- [179] J. J. P. Stewart, Application of the PM6 method to modeling the solid state, *J. Mol. Model.*, **14**, 499-535
- [180] Introduction to the eMinerals project, eMinerals, <http://www.eminerals.org/>, accessed on May 24, 2008
- [181] T. O. H. White, P. Murray-Rust, P. A. Couch, R. P. Tyler, R. P. Bruin, I. T. Todorov, D. J. Wilson, M. T. Dove, K. F. Austen, Application and Uses of CML within the eMinerals project, *Proceedings of the UK e-Science All Hands Meeting*, **2006**
- [182] Homepage, FoX (Fortran XML), <http://www.uszla.me.uk/FoX/>, accessed on May 24, 2008
- [183] CML for Computational Atomistics, CMLComp, <http://www.cmlcomp.org>, accessed on May 24, 2008
- [184] M. D. Segall, P. J. D. Lindan, M. J. Probert, C. J. Pickard, P. J. Hasnip, S. J. Clark, M. C. Payne, First-principles simulation: ideas,

- illustrations and the CASTEP code, *J. Phys. Cond. Matt.*, **14**, 2717-2743
- [185] MaterialsGrid: Large scale computer simulation of physical properties of materials, MaterialsGrid, <http://www.materialsgrid.org/>, accessed on May 24, 2008
  - [186] Homepage, Condor project, <http://www.cs.wisc.edu/condor/>, accessed on May 24, 2008
  - [187] Connecting Condor Pools with Flocking, Condor manual, [http://www.cs.wisc.edu/condor/manual/v6.8/5\\_2Connecting\\_Condor.html](http://www.cs.wisc.edu/condor/manual/v6.8/5_2Connecting_Condor.html), accessed on May 24, 2008
  - [188] Details of the UCC Condor pool, UCC, <http://www-ucc.ch.cam.ac.uk/C0/condor.html>, accessed on May 24, 2008
  - [189] M. Born, T. von Karman, Vibrations in Space Gratings (Molecular Frequencies), *Z. Physik.*, **13**, 297-309
  - [190] A. F. Block, *Z. Physik.*, **52**, 555
  - [191] The Cluster, MOPAC online manual, [http://openmopac.net/Manual/Solids\\_cluster.html](http://openmopac.net/Manual/Solids_cluster.html), accessed on May 24, 2008
  - [192] Considerations in Geometry Optimization, MOPAC online manual, [http://openmopac.net/Manual/Geometry\\_considerations.html](http://openmopac.net/Manual/Geometry_considerations.html), accessed on May 24, 2008
  - [193] O. Hassel, Structural Aspects of Interatomic Charge-Transfer Bonding, *Science*, **170**, 497-502
  - [194] I. Castellote, M. Moron, C. Burgos, J. Alvarez-Builla, A. Martin, P. Gomez-Sal, J. J. Vaquero, Reaction of imines with N-iodosuccinimide (NIS): unexpected formation of stable 1 : 1 complexes, *Chem. Commun.*, **2007**, 1281-1283



- [195] Homepage, American Mineralogist Crystal Structure Database, <http://rruff.geo.arizona.edu/AMS/amcsd.php>, accessed on May 24, 2008
- [196] Comparison of Structures of Crystalline Solids Predicted using PM6 with X-Ray, MOPAC, <http://openmopac.net/all.html>, accessed on May 24, 2008
- [197] Private communication with J. J. P. Stewart, author of the MOPAC program, July 2008
- [198] Bond-length histograms, CrystalEye, <http://wmm.ch.cam.ac.uk/crystaleye/bondlengths/index.html>, accessed on May 24, 2008
- [199] Error messages produced by MOPAC, MOPAC Online Manual, [http://openmopac.net/Manual/error\\_messages.html](http://openmopac.net/Manual/error_messages.html), accessed on May 24, 2008
- [200] J. Townsend, Automated Analysis and Validation of Chemical Literature, *PhD Thesis* **2007**, p137
- [201] Homepage, The R Project for Statistical Computing, <http://www.r-project.org/>, accessed on May 24, 2008
- [202] G. M. Day, W. D. S. Motherwell, H. L. Ammon, S. X. M. Boerrigter, R. G. Della Valle, E. Venuti, A. Dzyabchenko, J. D. Dunitz, B. Schweizer, B. P. van Eijck, P. Erk, J. C. Facelli, V. E. Bazterra, M. B. Ferraro, D. W. M. Hofmann, F. J. J. Leusen, C. Liang, C. C. Pantelides, P. G. Karamertzanis, S. L. Price, T. C. Lewis, H. Nowell, A. Torrisi, H. A. Scheraga, Y. A. Arnautova, M. U. Schmidt, P. Verwer, A third blind test of crystal structure prediction, *Acta Cryst.*, **B61**, 511-527
- [203] Private communication with J. J. P. Stewart, author of the MOPAC program, June 2008
- [204] C. Steinbeck, S. Krause, S. Kuhn, NMRShiftDB - Constructing a free chemical information system with open-source components, *J. Chem. Inf. Comput. Sci.*, **43**, 1733-1739

- [205] C. Steinbeck, S. Kuhn, NMRShiftDB - compound identification and structure elucidation support through a free community-built web database, *Phytochemistry*, **65**, 2711-2717
- [206] K. A. Blinov, Y. D. Smurnyy, M. E. Elyashberg, T. S. Churanova, M. Kvasha, C. Steinbeck, B. A. Lefebvre, A. J. Williams, Performance Validation of Neural Network Based  $^{13}\text{C}$  NMR Prediction Using a Publicly Available Data Source, *J. Chem. Inf. Model.*, **2008**, 48, 550-555
- [207] A. C. J. de Dios, Ab initio calculations of the NMR chemical shift, *Prog. Nucl. Magn. Reson. Spectrosc.*, **1996**, 29, 229
- [208] T. Helgaker, M. Jaszunski, K. Rudd, Ab Initio Methods for the Calculation of NMR Shielding and Indirect Spin-Spin Coupling Constants, *Chem. Rev.*, **99**, 293-352
- [209] R. J. Ditchfield, Molecular Orbital Theory of Magnetic Shielding and Magnetic Susceptibility, *J. Chem. Phys.*, **1972**, 56, 5688
- [210] K. Wolinski, J. F. Hinton, P. Pulay, Efficient Implementation of the Gauge-Independent Atomic Orbital Method for NMR Chemical Shift Calculations, *J. Am. Chem. Soc.*, **1990**, 112, 8251
- [211] J. R. Cheeseman, G. W. Trucks, T. A Keith, M. J. Frisch, A comparison of models for calculating nuclear magnetic resonance shielding tensors, *J. Chem. Phys.*, **1996**, 104, 5497
- [212] J. Casanovas, A. M. Namba, S. Leon, G. L. B. Aquino, G. V. J. da Silva, C. Aleman, Calculated and Experimental NMR Chemical Shifts of *p*-Menthane-3,9-diols. A Combination of Molecular Dynamics and Quantum Mechanics to Determine the Structure and Solvent Effects, *J. Org. Chem.*, **2001**, 66, 3775-3782
- [213] A. Baladina, V. Mamedov, X. Franck, B. Figadere, S. Latypov, Application of quantum chemical calculations of  $^{13}\text{C}$  NMR chemical shifts to quinoxaline structure determination, *Tet. Lett.*, **45**, 4003-4007

- [214] D. C. Braddock, H. S. Rzepa, Structural Reassignment of Obtusallenes V, VI, and VII by GIAO-Based Density Functional Prediction, *J. Nat. Prod.*, **2008**, 71, 728-730
- [215] S. D. Rychnovsky, Predicting NMR Spectra by Computational Methods: Structure Revision of Hexacyclinol, *Org. Lett.*, **8**, 13, 2895-2898
- [216] D. A. Forsyth, A. B. Sebag, Computed  $^{13}\text{C}$  NMR Chemical Shifts via Empirically Scaled GIAO Shieldings and Molecular Mechanics Geometries. Conformation and Configuration from  $^{13}\text{C}$  Shifts, *J. Am. Chem. Soc.*, **1997**, 119, 9483-9494
- [217] G. Barone, L. Gomez-Paloma, D. Duca, A. Silvestri, R. Riccio, G. Bifulco, Structure Validation of Natural Products by Quantum-Mechanical GIAO Calculations of  $^{13}\text{C}$  NMR Chemical Shifts, *Chem. Eur. J.*, **2002**, 8, 3233-3239
- [218] G. Barone, D. Duca, A. Silvestri, L. Gomez-Paloma, R. Riccio, G. Bifulco, Determination of the Relative Stereochemistry of Flexible Organic Compounds by Ab Initio Methods: Conformational Analysis and Boltzmann-Average GIAO  $^{13}\text{C}$  NMR Chemical Shifts, *Chem. Eur. J.*, **2002**, 8, 3240-3245
- [219] K. K. Baldridge, J. S. Siegel, Correlation of Empirical  $\delta(\text{TMS})$  and Absolute NMR Chemical Shifts Predicted by Ab Initio Computations, *J. Phys. Chem.*, **103**, 4038
- [220] D. D. Laws, H. Le, A. C. de Dios, R. H. Havlin, E. Oldfield, A Basis Size Dependence Study of Carbon-13 Nuclear Magnetic Resonance Spectroscopic Shielding and Alanyl and Valyl Fragments: Toward Protein Shielding Hypersurfaces, *J. Am. Chem. Soc.*, **117**, 9542
- [221] P. Cimino, L. Gomez-Paloma, D. Duca, R. Riccio, G. Bifulco, Comparison of different theory models and basis sets in the calculation of  $^{13}\text{C}$  NMR chemical shifts of natural products, *Magn. Reson. Chem.*, **2004**, 42, S26-S33

- [222] Private communication with Prof. H. S. Rzepa, September 2007
- [223] OpenContent License (OPL), OpenContent, <http://opencontent.org/opl.shtml>, accessed on June 15, 2008
- [224] JCAMP Standard NMR Data Format homepage, Acorn NMR Inc., <http://www.acornnmr.com/JCAMP.htm>, accessed on June 15, 2008
- [225] J-C Bradley, Open Notebook Science, Drexel COAS E-Learning, <http://drexel-coas-elearning.blogspot.com/2006/09/open-notebook-science.html>, accessed on June 15, 2008
- [226] Predicting  $^{13}\text{C}$  NMR shifts with Gaussian03, WWMM, <http://wwmm.ch.cam.ac.uk/data/nmr/>, accessed on June 15, 2008
- [227] E. S. Raymond, Release Early, Release Often, The Cathedral and the Bazaar, <http://catb.org/esr/writings/cathedral-bazaar/cathedral-bazaar/ar01s04.html>, accessed on June 15, 2008
- [228] Peter Murray-Rust, Homepage, A Scientist and the Web, <http://wwmm.ch.cam.ac.uk/blogs/murrayrust/>, accessed on June 15, 2008
- [229] M. Kaupp, O. L. Malkina, V. G. Malkin, Interpretation of  $^{13}\text{C}$  NMR chemical shifts in halomethyl cations. On the importance of spin-orbit coupling and electron correlation, *Chem. Phys. Lett.*, **265**, 55-59
- [230] O. L. Malkina, B. Schimmelpfennig, M. Kaupp, B. A. Hess, P. Chandra, U. Wahlgren, V. G. Malkin, Spin-orbit corrections to NMR shielding constants from density functional theory. How important are the two-electron terms?, *Chem. Phys. Lett.*, **296**, 93-104
- [231] What is NMRShiftDB?, NMRShiftDB, [http://nmrshiftdb.ice.mpg.de/portal/js\\_pane/P-Help;jsessionid=FBA908159F963A23D5BF8D06D7F1B830.tomcat2?URL=t1.html#whatis](http://nmrshiftdb.ice.mpg.de/portal/js_pane/P-Help;jsessionid=FBA908159F963A23D5BF8D06D7F1B830.tomcat2?URL=t1.html#whatis), accessed on June 15, 2008

- [232] Quality check of nmrshiftdb-data using CSEARCH-Algorithms, NMRPREDICT, [http://nmrpredict.orc.univie.ac.at/csearchlite/enjoy\\_its\\_free.html](http://nmrpredict.orc.univie.ac.at/csearchlite/enjoy_its_free.html), accessed on June 15, 2008
- [233] Peter Murray-Rust's blog, Comments for the entry titled 'Open NMR: Nick Day's final results', <http://wmm.ch.cam.ac.uk/blogs/murrayrust/?p=800#comments>, accessed on June 15, 2008
- [234] H. L. Morgan, The Generation of a Unique Machine Description for Chemical Structures - A Technique Developed at Chemical Abstracts Service, *J. Chem. Doc.*, **1965**, 5, 107-113
- [235] Cambridge Chemistry Department Crystallographic Repository, <http://wmm.ch.cam.ac.uk/projects/C3DeR/index.html>, accessed on November 20, 2008
- [236] Main Page, eCrystals Project Wiki, [http://wiki.ecrystals.chem.soton.ac.uk/index.php/Main\\_Page](http://wiki.ecrystals.chem.soton.ac.uk/index.php/Main_Page), accessed on November 20, 2008
- [237] C3DE (Cambridge CML Crystallographic Diagram Editor), <http://wmm.ch.cam.ac.uk/projects/C3DE/index.html>, accessed on November 20, 2008
- [238] How does InChI represent organometallic compounds? ,Unofficial InChI FAQ, <http://wmm.ch.cam.ac.uk/inchifaq/#How%20does%20InChI%20represent%20organometallic%20compounds?> , accessed on November 20,2008
- [239] S. J. Coles, N. E. Day, P. Murray-Rust, H. S. Rzepa and Y. Zhang, Enhancement of the chemical semantic web through the use of InChI identifiers, *Org. Biomol. Chem.*, **2005**, 3, 1832-1834
- [240] Dogpile homepage, <http://www.dogpile.com/>, accessed on May 27, 2008
- [241] Crystal Structure Report Archive, University of Southampton, <http://ecrystals.chem.soton.ac.uk/>, accessed on May 27, 2008

References are given in the style adopted by the Journal of Chemical Informatics and Computer Science.