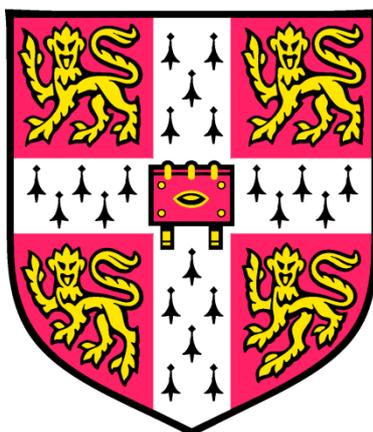


Information Extraction from Chemical Patents

David Matthew Jessop

Fitzwilliam College



This dissertation is submitted for the degree of Doctor of Philosophy

Preface

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

This dissertation does not exceed the word limit (60000) set by the Degree Committee

Abstract

Information Extraction from Chemical Patents

David Matthew Jessop

The automated extraction of semantic chemical data from the existing literature is demonstrated. For reasons of copyright, the work is focused on the patent literature, though the methods are expected to apply equally to other areas of the chemical literature.

Hearst Patterns are applied to the patent literature in order to discover hyponymic relations describing chemical species. The acquired relations are manually validated to determine the precision of the determined hypernyms (85.0%) and of the asserted hyponymic relations (94.3%). It is demonstrated that the system acquires relations that are not present in the ChEBI ontology, suggesting that it could function as a valuable aid to the ChEBI curators. The relations discovered by this process are formalised using the Web Ontology Language (OWL) to enable re-use.

PatentEye – an automated system for the extraction of reactions from chemical patents and their conversion to Chemical Markup Language (CML) – is presented. Chemical patents published by the European Patent Office over a ten-week period are used to demonstrate the capability of PatentEye – 4444 reactions are extracted with a precision of 78% and recall of 64% with regards to determining the identity and amount of reactants employed and an accuracy of 92% with regards to product identification. NMR spectra are extracted from the text using OSCAR3, which is developed to greatly increase recall. The resulting system is presented as a significant advancement towards the large-scale and automated extraction of high-quality reaction information.

Extended Polymer Markup Language (EPML), a CML dialect for the description of Markush structures as they are presented in the literature, is developed. Software to exemplify and to enable substructure searching of EPML documents is presented. Further work is recommended to refine the language and code to publication-quality before they are presented to the community.

Acknowledgments

I would like to thank Prof. Robert Glen and Prof. Peter Murray-Rust for supervision. I would also like to thank all those who have contributed to the creation of the software that has made this project possible – most notably Dr Peter Corbett for his work on OSCAR3, Dr Lezan Hawizy for her work on ChemicalTagger and Daniel Lowe for his work on OPSIN. Further thanks go to all those too numerous to name at the Unilever Centre, past and present, who have contributed to discussions and supported me in my work. Special thanks go to Dr Sam Adams, who volunteered to proof read this thesis, and to Jo for her love and support.

I am grateful to Unilever for funding.

Contents

Preface	i
Abstract.....	ii
Acknowledgments.....	iii
Contents.....	iv
List of Figures	vii
Glossary.....	ix
1. Introduction	1
1.1 Open and Closed Data	1
1.2 The Semantic Web	2
1.3 Semanticizing Chemistry.....	5
1.4 Information Extraction from Chemical Documents.....	5
1.5 Development Environment.....	7
2. Sources of Chemical Documents & Technologies for their Semantic Enrichment	8
2.1 Availability of Documents	8
2.1.1 Journal Articles.....	9
2.1.2 Theses	9
2.1.3 Patents	10
2.2 Key Technologies	14
2.2.1 XML & XPath	15
2.2.2 Regular Expressions	16
2.2.3 Machine-Understandable Chemical Formats	17
2.2.4 Chemical Markup Language.....	21
2.2.5 CMLXOM & JUMBO.....	25
2.2.6 OSCAR3	26
2.2.7 ChemicalTagger.....	39
2.2.8 OSRA.....	45
2.3 Conclusions	47
3. Representation and Manipulation of Markush Structures.....	48
3.1 Markush Structures.....	49
3.2 Polymer Markup Language	50
3.2.1 Representation of Polyethylene Oxide	50

3.2.2	Representation of Polystyrene	53
3.2.3	Representing Variability in PML.....	55
3.2.4	The Cambridge Polymer Builder	58
3.3	Extension of PML for Markush Structures	61
3.3.1	Frequency Variation.....	63
3.3.2	Homology Variation	63
3.3.3	Position Variation.....	66
3.3.4	Position and Count Variation	67
3.3.5	Inline Connection Tables.....	68
3.4	Building Representative Examples of a Markush Structure.....	70
3.5	Substructure Searching of Markush Structures	75
3.5.1	Implementing Extended Connection Tables	76
3.5.2	Building Extended Connection Tables.....	78
3.5.3	The Relaxation Algorithm.....	82
3.5.4	Examples	87
3.6	Conclusions	91
4.	Automatic Acquisition of Hyponymic Relations from the Chemical Literature	93
4.1	Hyponymic Relations	93
4.2	Hearst Patterns	94
4.2.1	OSCAR3 Implementation	95
4.3	Acquiring Hyponymic Relations	97
4.3.1	HearstFinder.....	98
4.3.2	Recording Hyponymic Relations	100
4.3.3	Content of the Derived Relations & Sources of Error	102
4.3.4	Trimming the Relations.....	105
4.3.5	HearstFinder Validation	107
4.4	Uses of Derived Data.....	116
4.4.1	Automatic Classification of Structural & Non-Structural Classes	117
4.4.2	Detection of Useful Relationships.....	120
4.4.3	Application to Data Searching.....	124
4.5	Conclusions	124
5.	High-Throughput Abstraction of Chemical Reactions – PatentEye	126
5.1	Downloading Patents.....	127
5.1.1	EPO Web Interface.....	127

5.1.2	Automated Downloading of EPO patents	128
5.1.3	Formation of the Patent Corpus	130
5.2	Document Enhancement	131
5.2.1	Paragraph Deflattening	132
5.2.2	Document Segmentation	133
5.2.3	Data Annotation.....	139
5.2.4	Experimental Paragraph Classification	154
5.2.5	Image Analysis.....	157
5.2.6	Back Reference Annotation	170
5.3	Extraction of Reactions	175
5.3.1	Conventional Format of Experimental Sections	176
5.3.2	Implementation of Automatic Reaction Extraction	177
5.3.3	Reaction Extraction Performance	195
5.4	Conclusions	198
6.	Results.....	199
6.1	Quality of Extracted Reactions.....	199
6.1.1	Corpus Formation	199
6.1.2	Product Validation	200
6.1.3	Reagent Validation.....	201
6.1.4	Spectra Validation.....	203
6.1.5	Automated Verification Validation	204
6.2	Enabling Reuse of the Extracted Data.....	209
7.	Conclusions	213
8.	Bibliography	216
	Appendix A	224
	Appendix B	227
	Appendix C	229
	Appendix D.....	230
	Appendix E	232

List of Figures

Figure 2-1: InChI representations of limonene	19
Figure 2-2: CML representation of acetaldehyde	22
Figure 2-3: Hydration of acetaldehyde	23
Figure 2-4: CML representation of a chemical reaction	24
Figure 2-5: OSCAR3 Architecture	26
Figure 2-6: Example inline document as produced by OSCAR3.....	32
Figure 2-7: OSCAR3 Data Annotations	37
Figure 2-8: ¹ H NMR regular expression.....	38
Figure 2-9: ChemicalTagger Architecture	40
Figure 2-10: Sample ChemicalTagger output.....	44
Figure 2-11: Example reaction scheme	45
Figure 3-1: Generic structure representing the monochlorinated toluenes	48
Figure 3-2: PML representation of poly(ethylene oxide).....	51
Figure 3-3: Atomistic representation for the g:o fragment	52
Figure 3-4: The creation of bonds between fragments in PML	53
Figure 3-5: PML representation of polystyrene.....	54
Figure 3-6: PML representation of a statistical copolymer.....	56
Figure 3-7: The front page of the Cambridge Polymer Builder.....	58
Figure 3-8: Designing a polymer	59
Figure 3-9: Results of polymer building	60
Figure 3-10: PML representation of the monochlorinated toluenes.....	62
Figure 3-11: Homology variation in EPML	64
Figure 3-12: Formal description of the alkoxy template	65
Figure 3-13: Position variation in EPML.....	66
Figure 3-14: Markush structure employing simultaneous position and count variation	67
Figure 3-15: Simultaneous position and count variation in EPML.....	68
Figure 3-16: Markush structure featuring variable cyclic unit.....	69
Figure 3-17: Inline connection tables in EPML.....	69
Figure 3-18: Example Markush structure	73
Figure 3-19: EPML representation of the example Markush structure	74
Figure 3-20: 3D (left) and 2D (right) views of a randomly-generated example compound	75
Figure 3-21: Superimposed structure representing the monochlorinated toluenes	75
Figure 3-22: Relaxation match of 3-aminopropanoyl chloride.....	84
Figure 3-23: Inconclusive results of relaxation matches	85
Figure 3-24: Example Markush structure (left) and corresponding ECT (right).....	88
Figure 4-1: Acquisition and Storage of Hearst Patterns.....	97
Figure 4-2: Grammatical structure of a Hearst Pattern	100
Figure 4-3: Distribution of Hearst Patterns across the patent corpus.....	102
Figure 4-4: Individual Hearst Pattern frequency across the patent corpus.....	103
Figure 4-5: The customised OSCAR3 ScrapBook.....	110

Figure 4-6: Annotated Hearst Pattern as produced by the OSCAR3 ScrapBook.....	111
Figure 4-7: Indirect ChEBI classification of acetone as a solvent.....	121
Figure 5-1: Search results for EP 1777210.....	127
Figure 5-2: Variation of downloaded and unique patents in the corpus.....	130
Figure 5-3: Identification of and Document Restructuring Using Consecutive Headings	138
Figure 5-4: Software architecture for the application of OSCAR3 data annotations to patent XML documents	140
Figure 5-5: Embedded images in the patent XML. The text has been shortened for the sake of brevity	158
Figure 5-6: EP1620437B1 Image 413	158
Figure 5-7: Types of images present in experimental sections.....	161
Figure 5-8: Input image (left) and correctly interpreted structure (right).....	164
Figure 5-9: Input image (top) and correctly interpreted structure (bottom).....	165
Figure 5-10: Input image (left) and correctly interpreted structure.....	165
Figure 5-11: Input image (top) and incorrectly interpreted structure (bottom).....	166
Figure 5-12: Input image (top) and unbuildable result (bottom)	167
Figure 5-13: Input image (top) and unbuildable result (bottom)	168
Figure 5-14: Runtime required for image analysis.....	169
Figure 5-15: Analogous reactions from EP1326865.....	170
Figure 5-16: Indexed and Tokenised Headings	172
Figure 5-17: Local annotation of sub-headings	174
Figure 5-18: EP1326865 - Example 79, Step 1	177
Figure 5-19: Abstracting reactions from patent text	178
Figure 5-20: Example NMR spectrum in CML	181
Figure 5-21: Sample ChemicalTagger markup of a reactant.....	182
Figure 5-22: ChemicalTagger output for mixed content	182
Figure 5-23: Automatically generated reactantList and spectatorList. For the sake of brevity, atom and bond elements have been removed	184
Figure 5-24: Identification of key reactants.....	188
Figure 5-25: Significant substructures in the analogous reaction	189
Figure 5-26: Proton environments in a non-trivial system	192
Figure 6-1: Sample RDF from the PatentEye Repository	211
Figure 6-2: Diagrammatic illustration of PatentEye Repository RDF	212

Glossary

- API** Application Programming Interface
- CAS** Chemical Abstracts Service
- ChEBI** Chemical Entities of Biological Interest
- CML** Chemical Markup Language
- DTD** Document Type Definition
- EPML** Extended Polymer Markup Language
- EPO** European Patent Office
- ECT** Extended Connection Table
- HTML** HyperText Markup Language
- InChI** IUPAC International Chemical Identifier
- JUMBO** Java Universal Molecular Browser for Objects
- JVM** Java Virtual Machine
- MEMM** Maximum Entropy Markov Model
- NLP** Natural Language Processing
- OCR** Optical Character Recognition
- OPSIN** Open Parser for Systematic IUPAC Nomenclature
- OSCAR** Open Source Chemistry Analysis Routines
- OSRA** Optical Structure Recognition Application
- OWL** Web Ontology Language
- MPT** Mean Pairwise Tanimoto Coefficient
- NMR** Nuclear Magnetic Resonance
- PDF** Portable Document Format
- PML** Polymer Markup Language
- RDF** Resource Description Framework
- SMILES** Simplified Molecular Input Line Entry Specification

URI Uniform Resource Indicator

USPTO United States Patent and Trademark Office

WIPO World Intellectual Property Organization

XML Extensible Markup Language

1. Introduction

Isaac Newton once wrote to Robert Hooke;

“If I have seen a little further it is by standing on the shoulders of giants”

This oft-quoted adage contains a great truth; in modern science, all new works are based upon something that has come before and so if we are to carry out new work, we must first *know* what has come before. In the modern age this can be difficult – research today is carried out on a huge scale, and a scientist searching for the answer to a simple question may find it impossible to find the appropriate needle in a vast, electronic haystack. Modern information systems give him at least a fighting chance, but it is by no means guaranteed that a computer system will contain the data he seeks, or have indexed the information it holds in sufficient depth to allow him to find his answer. This thesis seeks to address the problem of information flow in chemistry; the question of how to make information available to those who need it, when they need it.

1.1 Open and Closed Data

The scale of information output in modern chemistry is huge (1). The CAPLUS database (2) holds more than 32 million references to patents and journal articles and indexes more than 1500 current journals on a weekly basis, while the CAS REGISTRY (3) holds more than 54 million chemical compounds and the CASREACT (4) database more than 39 million single and multi-step reactions. But even these numbers do not do justice to the scale on which the research is conducted – inevitably, these databases will not be complete indexes of the published data, while published authors will limit the data they include in their documents to that which is directly relevant to their work, omitting much that they have generated during the course of it.

Much chemical information is not freely available – it may be locked to a paper format in a researcher’s lab book, effectively lost to the community, while the traditional business model of a journal requires the erection of paywalls. Such *closed data* obstructs the work of scientists, though they may not know it. Data may be closed with good reason – in commercial research, for example, revealing the detail of one’s work too early may risk the patentability of an invention and thereby its commercial value – but often data is closed that need not be.

The availability of data is vital for data-driven science such as spectra prediction and Quantitative Structure-Activity Relationships (QSAR), which has become increasingly important to the pharmaceutical industry as it seeks to control the spiralling costs of drug development. *Open data* – data that is freely available to the community – supports and enables such work. The more that the culture of open data spreads, the more such work becomes viable.

1.2 The Semantic Web

Tim Berners-Lee first described the concept of the Semantic Web (5). The idea is simple – the World Wide Web comprises a vast collection of information, but information that is largely meaningless to a computer. If it were to be made machine-understandable, then software agents could be developed that would be able use this information as a basis for reasoning and to make decisions. This concept, tied to that of open data, would allow for computerised scientists conducting their own data-driven research and reporting their conclusions back to humans. The concept of a machine performing research is not one for the world of science fiction – indeed, the robot scientist Adam has conducted its own hypothesis-driven research, reaching conclusions that were later validated by human researchers (6).

In order to make our information machine-understandable, it is necessary to formalise the semantics of the medium in which it is stored. For the semantic web, such formalisation is typically performed

by encoding the data using eXtensible Markup Language (XML). A bookshop, for example, might encode its catalogue as follows;

```
<catalogue>
  <book>
    <title>Of Mice and Men</title>
    <author>John Steinbeck</author>
    <ISBN>0141023570</ISBN>
    <price>£4.00</price>
  </book>
  <book>
    <title>War and Peace</title>
    <author>Leo Tolstoy</author>
    <ISBN>1853260622</ISBN>
    <price>£1.99</price>
  </book>
  ...
</catalogue>
```

In this example, the four pieces of information that are stored for each book – the title, author, ISBN number and price – are enclosed within appropriate XML tags. XML tags may be either opening, *e.g.* `<title>`, closing, *e.g.* `</title>` or empty, *e.g.* `<title />`. A computer may read this document and see that the `catalogue` contains a `book`, the `title` of which is “Of Mice and Men” and the `author` of which is “John Steinbeck”. By specifying the semantics in this way, we have made the data machine-understandable.

Of course, concepts of the same name can mean different things to different people – or even to the same person. The concept “book” may be a collection of bound pages to a publisher or a collection of bets placed by gamblers to a bookmaker. A “title” may be the name of a book or a prefix to a person’s name in polite conversation. In order to allow a machine to differentiate between different concepts of the same name, XML elements are assigned namespaces, for example;

```
<book xmlns="http://www.amazon.co.uk" />
```

The attribute `xmlns` on this `book` element defines the term “book” as having the meaning defined by Amazon. The namespaces used in XML are Internationalised Resource Indicators (IRIs), as

opposed to the more common Uniform Resource Locator (URL) – the difference being that an IRI may, but need not, point to the actual location of a resource and may contain characters chosen from a larger set. By using a unique namespace, an author may create his own XML vocabulary suitable for whatever task he has in mind. Concepts from differing namespaces may then be combined within the same document to perform the role required by the document’s author. This combination of flexibility and precision of concepts has led in recent years to XML becoming a *de facto* standard with, for example, XML-based formats being adopted by Microsoft Office and OpenOffice.

Though we may be some years away from computers routinely performing our science for us, some benefits of making our data machine-understandable are immediate – it is, for example, immediately made far more discoverable. Perhaps our frustrated researcher needs to know the glass transition temperature of polyvinyl alcohol. A text-based search on the web for the terms “glass transition temperature” and “polyvinyl alcohol” may be successful, or it may not. The glass transition temperature is often abbreviated as “ T_g ”, while polyvinyl alcohol is also known as poly(vinyl alcohol), poly(ethenol) and PVA – which is also the abbreviation for polyvinyl acetate. Moreover, the value of T_g for a polymer depends on the precise composition of the polymer and the method of measurement employed. The difficulties of locating key information in the literature have been previously noted (7). Our researcher would be greatly advantaged if he could define the property and substance of interest in terms that the machine can understand and allow the correct value to be discovered automatically from a set of precisely defined data.

1.3 Semanticizing Chemistry

The description of chemistry in terms understandable to a machine is supported by the XML dialect Chemical Markup Language (CML). CML allows for the description of atoms, molecules, spectra, reactions and more, and is discussed in section 2.2.4.

By embedding CML inside another XML document, it is possible to produce a document which is readable to humans and in which the chemistry may be understood by machines – a datument (8). This process may be carried out either by the original author, on the author's behalf by an editor during the process of document publication, or post-publication. Creation of XML is best-supported by the creation of authoring tools, such as the Microsoft Word plug-in Chem4Word which allows CML to be embedded directly into Word documents (9). Publishers are beginning to see the value of semantically enriching their output, such as in the Royal Society of Chemistry's Project Prospect (10; 11; 12), but there remains a vast amount of published chemical literature, both past and present, which is entirely unintelligible to a machine. This presents the central problem that this thesis seeks to address; to what extent can we identify and semantically enhance chemical information in published documents and extract it to form novel collections?

1.4 Information Extraction from Chemical Documents

Chemical documents take a variety of types – the most common being journal articles, patents and theses. Such documents are widely available, though terms of usage agreements may restrict the uses to which they may be put. They typically contain large amounts of chemical information, such as syntheses, characterisation data and properties of a wide range of chemical substances. Such information is typically structured in purely natural (*i.e.* human) language, which is manually scraped from the literature in order to populate chemical databases such as the aforementioned CAS systems at the cost of much time and effort.

The potential automation of this process offers two great advantages. Firstly, the much greater efficiency of an automated process will allow the creation of free data aggregation services such as CrystalEye (13) – enabling data-driven science, saving money for those who depend on the information and widening access to such information. Secondly, by reducing the marginal cost of data acquisition to a matter of machine time the potential scale on which the process operates will be widened. The goal is unquestionably worthwhile, but the technology is as yet too immature to fully supplant the human aspect.

Various systems have been developed to address specific aspects of the goal of information extraction from the chemical literature. In the 1980's, CAS developed an experimental system to aid in their work abstracting chemical reactions from the literature while the 1990's saw the development of optical structure recognition – software to identify and interpret chemical structure diagrams – which remains an active area of research today. Recent times have seen the development of the Open Source Chemistry Analysis Routines (OSCAR) and ChemicalTagger (14) toolkits in the Unilever Centre. OSCAR (15) allows for the semantic annotation of chemical documents, identifying chemical terminology and data within text, and is described in section 2.2.6. ChemicalTagger combines the chemical name recognition aspects of OSCAR with standard natural language parsing techniques to analyse the grammar of chemical texts as a precursor to machine understanding of the texts, and is described in section 2.2.7. These two toolkits between them perform key roles in the software developed as part of the current work and provide a platform for the development of large-scale systems for the liberation of chemical information from its containing documents.

1.5 Development Environment

The work for this thesis required the integration and development of a number of pre-existing open-source technologies such as OSCAR; XOM (16), a library for the manipulation of XML; CMLXOM (17), a library for the manipulation of CML and JUMBO (17), a library of CML-compatible cheminformatics tools. All of these libraries are written in the Java programming language – and so, for compatibility reasons, is the code that underlies the present work. It is hoped that, in time, this code will also be released under an open-source licence to allow its further use and development by third parties.

2. Sources of Chemical Documents & Technologies for their Semantic Enrichment

The current work is built upon two *sine qua non*. The first is that a suitable, and preferably large, set of chemical documents be found that may be used as source documents. The second is that, wherever possible, the relevant pre-existing tools and technologies must be employed. Accordingly, this chapter discusses the sources of chemical documents (section 2.1) and the existing technologies that are used in the current work (section 2.2) such as machine-understandable chemical formats, Chemical Markup Language and the software libraries that operate on it and the tools OSCAR3, ChemicalTagger and OSRA.

2.1 Availability of Documents

Crucial to the success of this work is the availability and usability of suitable source documents. Chemical documents are typically supplied in one of two types of formats; in a text format, or an image format. In an image format, the supplied data encodes an image of the document, rendering the text of the document not readily accessible. A computer must first perform Optical Character Recognition (OCR) before operations involving the text may occur. OCR technology is highly error-prone, and so documents that are supplied in an image format are to be avoided. Conversely, in a text format the text of the document is encoded using a character set – making it directly available to a computer program. Such documents are obviously preferable for the current work. XML offers an ideal format for text documents, allowing for the explicit definition of text formatting, document sections, *etc.* in an unambiguous manner.

The popular Portable Document Format (PDF) is notoriously difficult to work with. The format is designed to produce electronic copies of a document that describe its appearance rather than its

content. While allowing for the creation of documents that are quite suitable for displaying information to human users, PDF does not lend itself to the easy interpretation of the enclosed text. Indeed, the process of extracting text from a PDF has been compared to “converting hamburgers into cows” (18). Consequently, the PDF format is also to be avoided where possible.

2.1.1 Journal Articles

Perhaps the most familiar chemical document is the journal article. Short and focused upon a specific subject, journal articles are published frequently throughout an academic’s career. With the coming of the digital age, journal articles are widely available in digital formats from the journal’s website. Journal articles are most frequently supplied as PDF downloads and in an HTML format for direct viewing on the website. Since a journal’s publisher will frequently operate a business model that charges for access to its articles, the terms of use of the subscription will typically prohibit the automated downloading of documents that the current work requires. Though open-access journals, such as Acta Crystallographica Section E, exist they are very much atypical at present. The question of whether such arrangements should or will continue in the digital age is open and important, but beyond the scope of this thesis. So while journal articles constitute an important route for the communication of chemical information, the automated abstraction of information from journal articles was not attempted during the current work.

2.1.2 Theses

The standard route by which a PhD is gained requires the preparation of a thesis, and so a great number of chemical theses are produced around the world every year. After the conclusion of the examination process, a physical copy of the thesis is deposited with the university’s library and becomes a public document. It is curious that in the modern day there is not requirement for the

deposition of a digital copy of a thesis though the very great majority of PhD candidates will have prepared their thesis as a digital document. At the time of writing, the University of Cambridge's digital repository of scholarly works, DSpace@Cambridge (19), contains just three theses from the Department of Chemistry, suggesting a low take-up rate among candidates. The British Library operates the Electronic Theses Online Service (EThOS) (20) which offers access to those theses that the British Library has digitised or has been supplied with a digital copy of, though coverage is incomplete and not all of the UK's universities participate in the programme. Due to the difficulty of accessing a sufficient quantity of usable documents, theses were not considered a suitable source of documents for the current work.

2.1.3 Patents

In order to gain a patent, an inventor must disclose and describe the subject of his invention, and detail examples of the invention. In the field of chemistry, this frequently requires the claimant to describe the synthesis and characterisation of a number of example compounds, as well as to describe the background and subject of the invention. As a result, chemical patents contain a great deal of potentially useful information such as synthetic routes and compound properties.

In order to allow the public to know what has and what has not been patented, the documents are made public and, in the digital age, are widely accessible. Patent authorities and numerous other websites host copies of the documents. Though these documents are not supplied entirely without copyright protection, the restrictions are certainly less prohibitive than those that apply to journal articles. The United States Patent and Trademark Office, for example, permit a patent author to claim copyright over his work, provided that he allows for the facsimile reproduction of the original (21). The European Patent Office asserts copyright over the content of its website, but allows for its

adaptation and reuse without the need to acquire a licence subject to certain conditions (22). Patents therefore provide an ideal source of data for the current work.

2.1.3.1 EPO Patents

The European Patent Office (EPO) has begun publishing its patents in an XML format. This format uses standardised XML tags such as `heading` and `p` (paragraph) to define the formatting of the document, to indicate the location of images within the text as well as to link to a specified image file and for some elementary definition of sections of the documents. Crucially, these patents are published in a text format. The XML files may be downloaded from the EPO website (23), and are packaged into a ZIP file along with a PDF-formatted copy of the patent and a set of files corresponding to the images that are present in the patent. These images are supplied in the TIFF format and are given sequential file names that correspond to the image IDs that are used in the XML-formatted copy of the patent document, *e.g.* `imgb0006.tif`.

The content of the patent XML files is governed by a Document Type Definition (DTD) file that can be downloaded from the EPO website (24). The composition of the XML-formatted patents, whether used by convention or enforced by the DTD, is subsequently discussed.

Root Element

The root element of the XML documents is `ep-patent-document`. The common children of this element include `SDOBI`, `abstract`, `description`, `claims`, `ep-reference-list`. The only required child of `ep-patent-document` is `abstract`, although the other children mentioned will generally be present as well – `description`, for example, will in practice only be absent in those documents that do not contain a description of the invention *e.g.* patent search reports.

Alternatively, the children of `ep-patent-document` are permitted to be a series of `doc-page` elements, but this format is only employed when the pages of the application are included in an image format, and has not been encountered during the preparation of this thesis.

Abstract

The abstract element can be composed either of an `abst-problem` and an `abst-solution` element, or of one or more `p` elements. The `abst-problem` and `abst-solution` elements themselves consist of one or more `p` elements. The `p` (paragraph) elements contain text as well as formatting tags such as `br` and `sup` that perform the same roles as their namesakes in HTML, and further elements such as `tables`, `maths` and `chemistry` that enclose further content of a specific type.

SDOBI

The Sub-DOcument for Bibliographic (SDOBI) data uses proprietary tags to encode a wealth of metadata related to the patent, *e.g.* the tag `B110` contains the patent number, `B140` contains the date of publication of the patent and `B542` contains the title of the patent. The specification of these tags is contained in the patent DTD, but since this metadata is not used at any point in this thesis it shall not be further discussed.

Claims

By convention, each document contains three `claims` sections – one each in English, French and German. Each `claims` element contains one or more `claim` elements, and each `claim` element

contains one or more `claim-text` elements. A `claim-text` element is composed of text, HTML-style formatting tags and further tags in a similar manner as for the `p` element.

`ep-reference-list`

The `ep-reference-list` element contains one or more sets of a `heading` element followed by one or more `p` elements. The `heading` elements contain text and HTML-style formatting tags, and the `p` elements have been discussed previously.

Description

The `description` element contains the majority of the text of the patent, and the DTD allows it to be composed of one or more sets of a `heading` followed by a number of `p` elements. The DTD also allows for a number of elements to be used that correspond to well-defined sections of the patent document, *e.g.* `technical-field`, `industrial-applicability` and `description-of-embodiments` – unfortunately the comments in the DTD state that “these elements must NOT be used by contractors” and they do not occur in the patents that comprise the corpus used to prepare this thesis. As a result, the identification of the different sections of the patent documents is not the trivial task that the DTD allows for, and this task is discussed later.

2.1.3.2 Other Patents

Patents are, of course, published by organisations other than the EPO. The World Intellectual Property Organisation (WIPO) publishes patent documents through its website (25), primarily as images that are not suitable for the current work, though also as HTML-formatted text with minimal

markup of document sections. The United States Patent and Trademark Office (USPTO) produces text versions of its patents which are converted to XML and made available for bulk download via Google patents (26). At the time of writing the documents available for download date from 1976 to the present day, and are claimed to number approximately 7 million, across all subjects.

The format of the patent text within the USPTO XML files is similar to that used by the EPO, with `heading` and `p` elements containing the text of headings and paragraphs respectively, and these elements forming a single, unstructured list. References to external files, including images, ChemDraw and Mol files are present in the XML, and the supporting files are available as part of the bulk downloads.

Though the USPTO patents were not used in the current work it is believed that the techniques and technology developed for the EPO patents should be directly applicable to them, with the need for only some minor customisation. The set of USPTO patents would therefore constitute a second bulk set of source documents for a scaled-up version of the current work. The WIPO patents could also be used as source documents with the caveat that it would be necessary first to reformat the HTML versions of the patents and identify headings within the text before they could be subjected to the same process as those patents produced by the EPO and the USPTO.

2.2 Key Technologies

Of course, the work presented in this thesis has not been conducted in an intellectual vacuum. Much of it is built upon technologies that have been developed over the preceding years or decades, and the most important of these technologies are subsequently discussed.

2.2.1 XML & XPath

The basic terminology and format of XML, as well as the capacity it provides to render information machine-understandable, has already been discussed in section 1.3. Much of the functionality for writing and handling XML documents in the current work is provided by the XOM library (16). XOM provides such basic and essential tools as the ability to read, operate upon, and serialise XML documents while constantly ensuring that the document is *well-formed* – the requirement that, among other points, the document must contain a single *root element* from which all other elements in the document descend, and that the elements be correctly nested, *i.e.* they must not overlap. Key operations supported by XOM include the addition to and removal from the document of XML elements and attributes and of text content. XOM further supports the use of the XML query language XPath (27).

XPath is a means to select XML nodes (elements, attributes, *etc.*) from a document. The language permits the user to formulate simple, context-independent queries such as;

```
//molecule
```

which selects all elements named “molecule” that are descended from the starting node, the prefix “//” indicating that the position of the selected node in the document is unimportant, provided that it descends from the starting point. Queries may be more complex and involve context-dependent terms, for example;

```
/molecule//atom
```

which selects all elements named “atom” that are descended from an element named “molecule” that is itself a child of the starting node. The query may be further specified by the requirement that elements carry named attributes or that named attributes have specific values, for example;

```
/molecule[@name='benzene']//atom
```

which operates as for the previous example, with the added requirement that the `molecule` element must carry a `name` attribute with the value "benzene". The example XPath expressions given here are simple examples, but are sufficient to give an impression of the uses to which XPath is put in the current work.

2.2.2 Regular Expressions

Regular expressions ("regexes") are a powerful method for string matching, for which support has been added to a number of programming languages including Java. When using regular expressions, literal characters in the search regex match to characters in the target text, *e.g.* the regex "mol" matches the substring "mol" in each of the strings "mol", "molecule", "salbutamol" and "Smolensk". Certain characters are used as metacharacters and have roles other than to literally denote a character. For example, the metacharacter "\$" marks the end of a line, the metacharacters "(" and ")" denote the beginning and the end of a group respectively, and the metacharacter "?" notes that the preceding character or group is optional. Character classes may be used in regexes to match a single character from a well-defined set. Certain character classes are built in to the language, such as ".", which matches any character and "\s", which matches any whitespace character. Other character classes may be defined by the user by enclosing the permitted characters in square brackets, *e.g.* "[abc]" matches any one of the characters "a", "b" or "c". Further metacharacters permit iteration of the preceding character or group, such as "*", which defines that any number (including zero), and "+", which defines that one or more of the previous character or group must occur in a match. Regular expressions may also define lookaheads and lookbehinds, that require that a matching substring must be followed or preceded, respectively, by a specified regex, *e.g.* a lookahead is used in the regex "mol(=?\s)" to match "mol" in "salbutamol is used by asthmatics" but not in "Smolensk is in Russia".

Regular expressions may be used to match complex patterns of text and to extract this text from a document. In the current work, regular expressions are most notably used by OSCAR3 in order to match the highly stylised reports of spectral data that occur in chemical texts. This operation is discussed later, in section 2.2.6.5.

2.2.3 Machine-Understandable Chemical Formats

As the usage of computers to manage chemical information expanded, it became necessary to be able to represent chemical structures in a format that was interpretable by the machine. While the simple text strings “ethyl acetate” or “Oseltamivir” are easily understood by humans with sufficient domain knowledge, the chemical structures they represent cannot be trivially identified by a computer. Simple tasks such as substructure searching or calculation of molecular weights therefore cannot be automated if the input to the program is not machine-understandable.

Machine-understandable chemical formats have proliferated over the years – the open-source format-conversion tool Open Babel supports more than 90 such formats (28). Two simple formats – SMILES and InChI – are employed in the current work and are subsequently discussed.

2.2.3.1 SMILES

Simplified Molecular Input Line Entry Specification (SMILES) (29), is a popular form of line notation – a method for representing chemical structures in which a single string encodes the structure. In SMILES, atoms are represented using the abbreviated forms of their chemical elements. Atoms may be indicated to be bonded to one another when their element symbols are adjacent to one another, and bonds are assumed to be single bonds unless the two symbols are connected by the symbol = or #, marking double and triple bonds respectively, and unless the atoms are from the limited subset

permitted to be marked as aromatic by using their lowercased element symbol, *e.g.* “n” and “c”. To keep SMILES strings readable, it is assumed that hydrogen atoms are present in sufficient number and appropriate positions to occupy free valencies. Thus, propanal may be represented by the SMILES string “CCC=O”.

Of course, not all chemical structures take the topological form of lines. Branches in a molecule may be indicated by enclosing the side chain within brackets, while ring closures may be indicated by following the two atoms between which a bond is present with the same number. Thus, 1,1-dimethylcyclohexane may be represented by the SMILES string “CC1(C)CCCCC1”.

SMILES strings provide a concise means of representing chemical structures in a machine-understandable format, and have the advantage that it is possible to produce representations for simple structures such as those above that are comprehensible to humans. SMILES strings have the disadvantage, however, that it is possible to represent a single chemical structure with a number of different SMILES strings. Propanal, for example, may be represented as “CCC=O” as above, as “O=CCC” or as “C(C=O)C” among many other permutations. It is therefore not possible to determine if the structure represented by two SMILES strings is the same by simple text comparison of the two strings. Though various proprietary algorithms exist for the canonicalization of SMILES strings, such as CANGEN (30), the differing algorithms produce differing canonical SMILES strings for the same connection table. Consequently, recent attention has focused on an open standard – the International Chemical Identifier (InChI).

2.2.3.2 InChI

The InChI technical manual (31) states that;

“The objective of the Identifier is to provide a string of characters capable of uniquely representing a chemical compound... Since InChI is intended to serve as a precise digital signature of a compound, it must have two properties: 1) different compounds

(as defined by their 'connection tables') must have different identifiers and 2) a single compound must have a single identifier, regardless how its structure is drawn."

The InChI (32; 33) represents a chemical structure in a series of layers and sub-layers. Sub-layers are indicated by preceding their content with the string "/?" where "?" is a single character code that identifies the sub-layer. The connection layer, indicating connectivity of the atoms in the molecular graph, for example, is indicated by the string "/c". Sub-layers are present only when required to describe the structure the InChI represents. For example, consider the InChIs for limonene, (R)-limonene and (S)-limonene as shown in Figure 2-1.

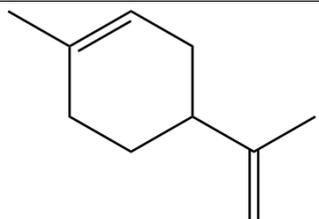
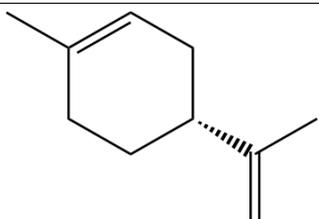
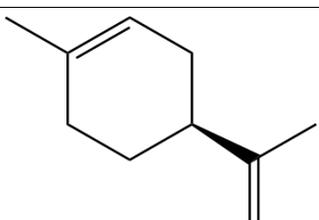
 <p>Limonene</p>	<p>InChI=1/C10H16/c1-8(2)10-6-4-9(3)5-7-10/h4,10H,1,5-7H2,2-3H3</p>
 <p>(R)-limonene</p>	<p>InChI=1/C10H16/c1-8(2)10-6-4-9(3)5-7-10/h4,10H,1,5-7H2,2-3H3/t10-/m0/s1</p>
 <p>(S)-limonene</p>	<p>InChI=1/C10H16/c1-8(2)10-6-4-9(3)5-7-10/h4,10H,1,5-7H2,2-3H3/t10-/m1/s1</p>

Figure 2-1: InChI representations of limonene

The InChI for limonene is composed as follows;

- InChI=1 – a declaration stating that the InChI is version 1
- /C10H16 – the molecular formula sub-layer (prefixed “/”), stating that the molecular formula is C₁₀H₁₆
- /c1-8(2)10-6-4-9(3)5-7-10 – the connectivity layer (prefixed “/c”), defining connections between atoms in the molecular graph; in this case atom 1 is connected to atom 8, which is connected to atoms 2 and 10 *etc.*
- /h4,10H,1,5-7H2,2-3H3 – the hydrogen layer (prefixed (“/h), defining the positions of hydrogen atoms in the molecule; in this case, atoms 4 and 10 have one hydrogen atom each, atoms 1 and 5-7 have two hydrogen atoms each *etc.*

When the stereochemistry of the chiral centre is specified as (R), the InChI is extended as follows;

- /t10- – the sp³ stereo sub-layer; in this case indicating that atom 10 has stereochemistry of parity “-” according to the InChI algorithm
- /m0 – indicating whether all defined sp³ stereochemistry should be inverted, allowing enantiomers of molecules with multiple stereocentres to share an identical sp³ stereo sub-layer
- /s1 – defining the type of stereochemistry as absolute or relative; in this case, absolute

When the InChI for (S)-limonene is calculated, it can be seen to be identical to that of (R)-limonene with the exception that the “/m0” becomes “/m1”, indicating the inversion of the stereocentre.

In order to produce canonical representations, the ordering of layers in an InChI is mandated. As a result, InChIs for related structures will start identically – for example, InChIs for all the isomers of limonene will start “InChI=1/C10H16”, while all InChIs for all the stereoisomers of limonene will start “InChI=1/C10H16/c1-8(2)10-6-4-9(3)5-7-10/h4,10H,1,5-7H2,2-3H3”, as seen above. The content of the various sub-layers is assured to be canonical by various procedures as detailed in the InChI technical manual; for example the molecular formulae are formatted according to the Hill system (34).

The generation of InChIs is supported by a number of popular software packages such as ChemDraw (35) and OpenBabel (28). Functionality for the automatic generation of InChIs from within Java programs is provided by the JNI-InChI library (36) and it is this library to which JUMBO delegates much of the process. Consequently, InChIs are a convenient canonical identifier for small molecules and are used where appropriate throughout the current work.

2.2.4 Chemical Markup Language

Chemical Markup Language (CML) is an XML-based language for the description of chemistry (37; 38; 39; 40; 41). As such, CML allows for the description of machine-understandable connection tables and much more besides. For example, the connection table of acetaldehyde may be represented as in Figure 2-2.

```

<molecule xmlns="http://www.xml-cml.org/schema" id="m1">
  <name dictRef="nameDict:unknown">acetaldehyde</name>
  <atomArray>
    <atom id="a1" elementType="C" />
    <atom id="a2" elementType="C" />
    <atom id="a4" elementType="O" />
    <atom id="a5" elementType="H" />
    <atom id="a6" elementType="H" />
    <atom id="a7" elementType="H" />
    <atom id="a8" elementType="H" />
  </atomArray>
  <bondArray>
    <bond id="a1_a2" atomRefs2="a1 a2" order="S" />
    <bond id="a1_a4" atomRefs2="a1 a4" order="D" />
    <bond id="a1_a5" atomRefs2="a1 a5" order="S" />
    <bond id="a2_a6" atomRefs2="a2 a6" order="S" />
    <bond id="a2_a7" atomRefs2="a2 a7" order="S" />
    <bond id="a2_a8" atomRefs2="a2 a8" order="S" />
  </bondArray>
</molecule>

```

Figure 2-2: CML representation of acetaldehyde

The individual atoms that make up the molecule are represented by the `atom` elements, which are contained within the `atomArray` element. Bonds are represented by the `bond` elements, contained within the `bondArray` element. The chemical element of each of the atoms is defined by the `elementType` attribute on the `atom` element, while the bond order is specified by the `order` attribute on the `bond` element and the two (assuming a conventional two-centre bond) atoms between which the bond exists are identified by the `atomRefs2` attribute on the `bond` element – the value being a concatenation of the unique ids of the atoms between which the bond exists. The example above does not include any information that could not be encoded in most if not all machine-understandable formats, but the great advantage of CML is that it provides a platform for the description of chemical information that is more complex than simply molecular connectivity. For example, CML vocabularies exist for the description of chemical reactions (42) and spectral data (43). Furthermore, these vocabularies may co-exist within the same document rather than require, for example, one document describing a molecular structure to link to a separate document that describes its NMR spectrum. For example, the hydration of acetaldehyde, as shown in Figure 2-3,

may be represented in CML as in Figure 2-4, in which the connection tables of the reaction components have been omitted for the sake of brevity.



Figure 2-3: Hydration of acetaldehyde

```

<reaction xmlns="http://www.xml-cml.org/schema">
  <reactantList>
    <reactant>
      <molecule id="m1">
        <name dictRef="nameDict:unknown">acetaldehyde</name>
      </molecule>
    </reactant>
    <reactant>
      <molecule id="m2">
        <name dictRef="nameDict:unknown">water</name>
      </molecule>
    </reactant>
  </reactantList>
  <spectatorList>
    <spectator>
      <molecule id="m3">
        <name dictRef="nameDict:unknown">
          hydrogen chloride
        </name>
      </molecule>
    </spectator>
  </spectatorList>
  <productList>
    <product>
      <molecule id="m4">
        <name dictRef="nameDict:unknown">
          1,1-ethanediol
        </name>
        <spectrum type="hnmr">
          <peakList>
            <peak xValue="1.40" integral="3.0"
              yUnits="unit:hydrogen"
              peakMultiplicity="cmlx:doublet">
              <peakStructure type="coupling" value="6.8"
                units="unit:hz" />
            </peak>
            <peak xValue="3.65" integral="2.0"
              yUnits="unit:hydrogen"
              peakMultiplicity="cmlx:singlet" />
            <peak xValue="5.13" integral="1.0"
              yUnits="unit:hydrogen"
              peakMultiplicity="cmlx:quartet">
              <peakStructure type="coupling" value="6.8"
                units="unit:hz" />
            </peak>
          </peakList>
        </spectrum>
      </molecule>
    </product>
  </productList>
</reaction>

```

Figure 2-4: CML representation of a chemical reaction

It can be seen that the reaction is described by its component reactant, spectator (*e.g.* solvents and catalysts) and product molecules. The ^1H NMR spectrum of the product molecule is also contained within the document, being present as a `spectrum` child of the `product` molecule.

The examples given above demonstrate a small but, in the context of this thesis, significant subset of the capacity of CML to define chemical information. By rendering chemical information machine-understandable, CML allows for the creation of systems that integrate data of a variety of types and from a variety of sources to perform novel research – a semantic web for chemistry.

2.2.5 CMLXOM & JUMBO

Since the purpose of Chemical Markup Language is to allow computers to understand chemistry, it follows that there is a need for tools for reading, writing and manipulating CML to allow programmers to implement CML-compliant software. These roles are filled by the open-source libraries CMLXOM and JUMBO (44). CMLXOM is intended as a CML-specific extension to the popular open-source XML manipulation library XOM (16), while JUMBO serves as an open-source collection of CML-compliant cheminformatics utilities. Further JUMBO-related projects exist to provide specific pieces of functionality, such as the JUMBOConverters (45) project that provide conversion between CML and a number of other formats. While the great majority of code development that was undertaken as part of the current work occurred in projects specific to the task at hand, functionality already provided as part of CMLXOM and JUMBO was reused where possible and functionality likely to be of use to others was added to the shared libraries.

The current work used versions 5.5.1 of JUMBO, 2.5.1 of CMLXOM, 0.3 of JUMBOConverters and 1.2.3 of XOM.

2.2.6 OSCAR3

The OSCAR project has, over the last few years, resulted in the creation of a number of tools for the analysis of chemical texts. OSCAR3 (46) is the latest incarnation of this project, and is a tool for the automated annotation and semantic enrichment of chemical texts. The primary function of OSCAR3 is to recognise chemical named entities and, where possible, to resolve chemical names to their corresponding connection tables. Where OSCAR3 recognises a named entity, annotations may be created in two forms – standoff, where the annotations are held in a separate document, and inline, where the annotations are directly embedded in the document under analysis. The system architecture of OSCAR3 is summarised in Figure 2-5, and the functioning of the program is subsequently discussed.

Where not otherwise noted, the version of OSCAR3 used in the current work is revision 877, dated 2010-07-21.

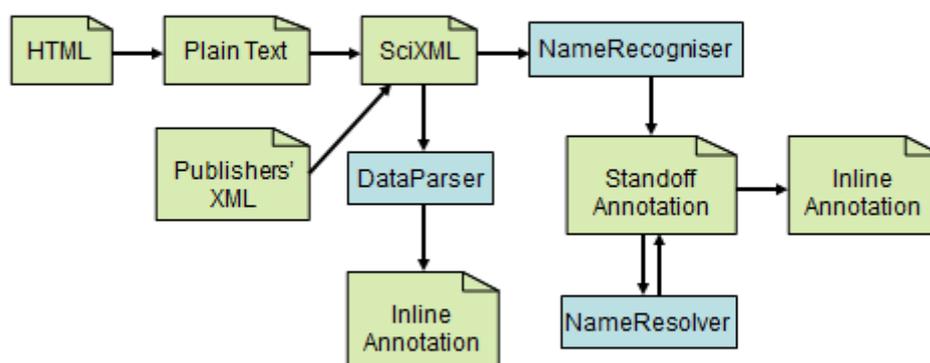


Figure 2-5: OSCAR3 Architecture

2.2.6.1 Input

OSCAR3 accepts input in a variety of formats. The system architecture is strongly dependent upon the SciXML format – much of the code relies upon the subject document being held in this format. As an initial step, other input formats are converted to SciXML and tools exist within OSCAR3 for this conversion to be effected from plain text and from other XML formats, and for the conversion of HTML to plain text from which the SciXML document may be created, as shown in Figure 2-5.

2.2.6.2 Key OSCAR3 Resources

The various stages of processing implemented in the OSCAR3 workflow make use of a number of key resources that are shared by the different modules. These resources provide examples of chemical and non-chemical terminology and associated structures. The resources are subsequently outlined.

ChemNameDict

ChemNameDict is a dictionary of chemical names that is largely derived from the ChEBI Ontology (47; 48) as at August 2009. ChemNameDict contains records for 13068 unique compounds, and each record contains precisely one InChI, at most one SMILES string and at least one chemical name. The large majority of the records that are derived from the ChEBI Ontology also contain at least one ChEBI ID, while the 52 that have been manually added by the OSCAR3 developers and were not present in the ChEBI Ontology do not.

Ontology.txt

Ontology.txt provides a list of terms from the ChEBI, FIX (49) and REX (50) ontologies and their corresponding ontology IDs.

Lexicon

The Lexicon provides a list of chemical elements, both names and atomic symbols, from Hydrogen to Darmstadtium, element 110.

ChemAses & NonChemAses

ChemAses comprises a list of the names of 924 enzymes that are intended to be recognised by OSCAR3 – those of named entity type ASE, as described in section 2.2.6.3 – that was manually collected from the Gene Ontology (GO) (51), while NonChemAses is a list of 126 enzyme names from GO that are not of type ASE, *e.g.* “epimerase” and “transferase”.

UsrDictWords

UsrDictWords is a list of approximately 96000 English words, included to provide as wide a variety of non-chemical words as possible. This dictionary contains chemical terms commonly used in English speech, such as “aspirin” and “benzene”. Terms from UsrDictWords are therefore only taken as examples of non-chemical terms if they do not also appear in one of the lists of chemical terms.

Further Training Data

Further training data is provided from a number of chemical documents in which named entities have been hand annotated by the OSCAR3 developers. This data includes the lists of tokens that have occurred in the texts and their state as chemical or non-chemical terms.

2.2.6.3 Named Entity Recognition

The key function of OSCAR3 is the recognition of named entities. A number of classes of named entities are recognised by OSCAR3. These classes are fully described in the OSCAR3 annotation guidelines, and can be summarised as follows;

- Chemical (CM) – used for terms that denote a specific chemical structure, substructure or generic structure *e.g.* “water”, “methyl” or “ketone”. The CM class is not applied to terms that describe function or properties, such as “ligand” or “base”.
- Reaction (RN) – used for names of reactions that are derived from terms of type CM or named after chemists, *e.g.* “ozonolysis” – derived from “ozone” – or “Hofmann degradation” – named after the 19th century chemist August Wilhem von Hofmann. RNs may occur in any part of speech *e.g.* “hydrolyse” or “methylated”.
- Chemical Adjective (CJ) – used for adjectives that are derived from words of type CM and do not occur as part of a CM, *e.g.* “aqueous” or “citric” in “citric acidity”.
- Enzyme (ASE) – used for enzyme names that are derived from words of type CM or RN, *e.g.* “beta-lactamase” or “hydrolase”.
- Chemical Prefix (CPR) – used for prefixes that would normally occur as part of a CM but are not being used as part of one, *e.g.* “cis-” in “cis- isomer” or “1,3-” in “1,3-dipolar”.

- Ontology term (ONT) – used for a set of dictionary terms that do not belong to any of the above types but are of sufficient importance to have been included in the ontology.txt resource, *e.g.* “acid” or “base”.

Previous work has involved the manual annotation of a corpus according to the OSCAR3 annotation guidelines by separate human annotators and has demonstrated inter-annotator agreement of around 90% (52). The named entities classes are further divided into subclasses. The assignment of named entities to subclasses is not typically of importance to the user and does not form a part of the standard process of named entity recognition. Indeed, the performance of automatic subclass assignment is poor compared to that of the named entity recognition (53). Each of the classes CM, RN, CJ and ASE are divided into subclasses while the classes CPR and ONT are not.

The class CM has three major and three minor subclasses. The major subclasses are EXACT, for specific compounds such as “acetic acid”, CLASS, for classes of compounds such as “carboxylic acid” and PART, for parts of specific compounds and classes thereof such as “carbonyl” and “alkyl”. The three minor subclasses are intended for terms that do not easily fit into one of the three major subclasses and are SPECIES, for elements used in the sense of “atmospheric carbon”, SURFACE and POLYMER, used for surfaces and polymers respectively. The subclasses of RN are REACT, for specific named reactions such as “methylation”, DESC, for descriptions of compounds such as in “the chlorinated solvent” and MOVE, for words that denote the movement of chemical compounds such as in “the chlorinated swimming pool”. The subclasses of CJ are EXACT, CLASS and PART, as for CM, plus ACID, for adjectives referring to an acid such as “citric”, SOLUTION, for adjectives referring to solutions such as “aqueous” and RECEPTOR for adjectives that refer to receptors such as “nicotinic”. The subclasses of ASE are not relevant to the current work and are detailed in the literature for the interested reader.

Once named entity recognition has been performed, the results are used to create a set of standoff annotations. If desired by the user, these standoff annotations are then passed to the

NameResolver class (see section 2.2.6.4) in order to generate and incorporate connection tables for named entities of type CM where possible. Finally, the set of standoff annotations may be used to create an *inline document*, embedding named entity annotations into the subject document. For example, the input text “Methyl bromide is my favourite alkyl halide.” is rendered as shown in Figure 2-6.

```

<PAPER>
  <BODY>
    <DIV>
      <HEADER />
      <P>
        <ne id="o1" surface="Methyl bromide" type="CM"
          confidence="0.8851147923587711"
          SMILES="[H]C([H])([H])Br"
          InChI="InChI=1/CH3Br/c1-2/h1H3" cmlRef="cml1"
          ontIDs="CHEBI:39275">
          Methyl bromide
        </ne>
        is my favourite
        <ne id="o2" surface="alkyl halide" type="CM"
          confidence="0.8596820024542137" rightPunct="."
          ontIDs="CHEBI:24469">
          alkyl halide
        </ne>
        .
      </P>
    </DIV>
  </BODY>
  <cmlPile>
    <cml convention="cmlDict:cmlite" id="cml1"
      xmlns="http://www.xml-cml.org/schema"
      xmlns:cmlDict="http://www.xml-cml.org/dictionary/cml/"
      xmlns:nameDict="http://www.xml-cml.org/dictionary/cml/name/">
      <molecule id="m1">
        <name>Methyl bromide</name>
        <identifier convention="iupac:inchi">
          InChI=1/CH3Br/c1-2/h1H3
        </identifier>
        <name dictRef="nameDict:unknown">Methyl bromide</name>
        <atomArray>
          <atom id="a1" elementType="C" />
          <atom id="a3" elementType="H" />
          <atom id="a4" elementType="H" />
          <atom id="a5" elementType="H" />
          <atom id="a2" elementType="Br" hydrogenCount="0" />
        </atomArray>
        <bondArray>
          <bond id="a1_a3" atomRefs2="a1 a3" order="S" />
          <bond id="a1_a4" atomRefs2="a1 a4" order="S" />
          <bond id="a1_a5" atomRefs2="a1 a5" order="S" />
          <bond id="a2_a1" atomRefs2="a2 a1" order="S" />
        </bondArray>
      </molecule>
    </cml>
  </cmlPile>
</PAPER>

```

Figure 2-6: Example inline document as produced by OSCAR3

OSCAR3 takes a modularised approach to named entity recognition and implements an extensible architecture. Two approaches are currently implemented – the PatternRecogniser and the MEMMRecogniser. Both approaches operate by breaking an input string of text into *tokens* –

continuous sets of characters that may or may not correspond to words. The `PatternRecogniser` and `MEMMRecogniser` attempt to identify within the resulting token sets any examples of the named entity classes described above.

`PatternRecogniser`

The strategy employed by the `PatternRecogniser` attempts to classify tokens as chemical or non-chemical before joining sequential tokens into multi-token named entities where appropriate. The classification is carried out by comparing the token to the lists of chemical and non-chemical tokens derived from the OSCAR3 resources. If the token is contained in a list of chemical tokens derived from the resources, it is known to be chemical, while if it is not contained in a chemical list but is contained in the `UsrDictWords` resource then it is known to be non-chemical. If the token is previously unseen then a naïve Bayesian model is used to classify the token. This model is trained using 4-grams – sequential substrings of the token of length 4 – of the chemical and non-chemical terms in the OSCAR3 resources. For example, for the token “pyridine” the 4-grams would be “^pyr pyri yrid ridi idin dine ine\$”. The first and last 4-grams here indicate specifically that the characters “pyr” and “ine” were found at the beginning and the end of the string respectively. Once trained, the model uses the 4-grams of an unseen token to predict the probability that it belongs to the chemical class, and if this probability exceeds a user-modifiable threshold then the token is classified as chemical.

Once the tokens have been determined to be chemical or non-chemical, a pre-classifier then putatively assigns each of the chemical tokens to one of the named entity classes, based on the ending of the token. For example, tokens ending in “ic” are putatively assigned as CJ, with the exception of “arsenic” which is putatively assigned as CM. Adjacent chemical tokens are then joined together into single named entities where appropriate. For example, “ethyl acetate” should be

recognised as a single named entity, whereas “carbonyl carbon” should not. This process is controlled by regular expression style patterns based upon common naming conventions such as “*yl *ate”, “*ium *ide” or “*ic acid”. Each of these patterns defines the named entity class to which the term should be assigned, for example terms of the form “CPR and RN” such as “ α - and β -methylation” are assigned as RN. Those tokens that have not been combined to form a multi-token name are then confirmed as members of their putative class. When identifying named entities using the `PatternRecogniser`, the assignment of named entity subclasses is not supported.

MEMMRecogniser

A second `NameRecogniser` implementation, the `MEMMRecogniser` (54), replaces the regular expressions employed by the `PatternRecogniser` with a Maximum Entropy Markov Model (MEMM), using the implementation supplied in the OpenNLP package `Maxent` (55). The pre-classification of tokens is carried out as described above, then for each token a number of features are generated. Such features may describe the token itself, such as the named entity class assigned by the pre-classifier and the 4-grams used in the pre-classification procedure, or contextual information about the usage of the token such as the tokens that surround the token in question. These features are then used by a rescorer to identify named entities. The rescorer consists of four individual maximum entropy classifiers, one for each of the classes CM, RN, CJ and ASE. Each of these four classifiers, trained on data extracted from manually annotated documents, predicts the probability that a given term is a named entity of the class for which the classifier has been trained and the most likely named entity subclass. Named entities are then identified within the text by discounting “blocked” entities – those that overlap with another entity with a higher confidence score. In the term “ethyl acetate”, for example, the entities “ethyl” and “acetate” would be considered individual named entities with low confidence scores, while the named entity “ethyl

acetate” would have a far higher confidence score and therefore supersede the single-token named entities.

The MEMMRecogniser is considered to provide higher performance than the PatternRecogniser and is the NameRecogniser that OSCAR3 uses by default.

2.2.6.4 NameResolver

The function of the NameResolver is, where possible, to generate a connection table for a named entity in InChI, SMILES and CML formats. In a conventional OSCAR3 workflow, the NameResolver modifies the set of stand-off annotations that has been produced by the NameRecogniser to include the connection tables, though as part of the work done in preparation of this thesis its capability was widened to allow its use as part of a library and the resolution of individual chemical names. Consequently, it is now possible to use the NameResolver to resolve individual chemical names to structures without the need to create and parse a SciXML document.

When attempting to resolve a name, a number of strategies are employed. First, if the named entity is found in the NameResolver cache – the set of terms previously resolved for the document – then the previously generated results are used. Secondly, the term is looked up in the Lexicon and ChemNameDict resources and, if the name being queried is found, the connection tables stored therein are returned. Finally, the name is passed to the Open Parser for Systematic IUPAC Nomenclature (OPSIN). The results generated at any stage are cached to facilitate more rapid functioning of the code.

The differing methods for resolving names produce connection tables in various formats. OPSIN, for example, generates a CML representation of the molecule, while a name resolved from ChemNameDict will have been resolved to InChI and, most to likely, to SMILES. The formats to which the name has not been resolved are generated from the known formats and the three formats are

added to the document being processed, in a conventional OSCAR3 workflow, or are returned to the calling function if OSCAR3 is being used as an external library.

2.2.6.5 DataParser

OSCAR3 also has a secondary but highly useful function implemented in the DataParser class – the capacity to recognise and annotate sections of analytical data such as NMR and mass spectra by using regular expressions to match the highly stylised formats authors use when reporting them. This functionality was the origin of the OSCAR project, which began life as the Experimental Data Checker – a tool to allow authors of papers to be published in RSC journals to check that their experimental sections both conform to the journal’s style guidelines and contain no identifiable errors (56; 57). The functionality has been retained within the OSCAR3 project, though is not integrated with the name recognition and resolution functionality. As shown in Figure 2-5, the DataParser produces an inline annotation document of its own, in which recognised chunks of data are annotated but chemical names are not, for example the annotations generated for the input text “Recorded spectrum of ethyl acetate: 1H NMR(400MHz) 1.20 (3H, t), 1.97 (3H, s), 4.10 (2H, q)” are shown in Figure 2-7.

```
<PAPER>
  <BODY>
    <DIV>
      <HEADER />
      <P>
        Recorded spectrum of ethyl acetate:
        <spectrum type="hnmr">
          1H NMR (
            <quantity type="frequency">
              <value>
                <point>400</point>
              </value>
              <units>MHz</units>
            </quantity>
          )
          <peaks type="..">
            <peak>
              <quantity type="shift">
                <value>
```

```

        <point>1.20</point>
    </value>
</quantity>
(
<quantity type="integral">
    <value>
        <point>3</point>
    </value>
    <units>H</units>
</quantity>
,
<quantity type="peaktype">t</quantity>
)
</peak>
,
<peak>
    <quantity type="shift">
        <value>
            <point>1.97</point>
        </value>
    </quantity>
    (
    <quantity type="integral">
        <value>
            <point>3</point>
        </value>
        <units>H</units>
    </quantity>
    ,
    <quantity type="peaktype">s</quantity>
    )
</peak>
,
<peak>
    <quantity type="shift">
        <value>
            <point>4.10</point>
        </value>
    </quantity>
    (
    <quantity type="integral">
        <value>
            <point>2</point>
        </value>
        <units>H</units>
    </quantity>
    <quantity type="peaktype">q</quantity>
    )
)
</peak>
</peaks>
</spectrum>
</P>
</DIV>
</BODY>
</PAPER>

```

Figure 2-7: OSCAR3 Data Annotations

It can be seen that the annotations that have been applied to the text by OSCAR3 have created a machine-understandable spectrum, in which the peak shifts, integrals and multiplicities have been individually marked up, allowing for a machine that recognises the format of the OSCAR3 data annotations to read the spectrum. A converter exists within the JUMBOConverters (45) project to transform the output thus produced by OSCAR3 for NMR spectra into Chemical Markup Language, facilitating the easy reuse of the data.

The regular expressions that form the core of this module are, for ease of comprehension and reuse, formatted in XML and implemented as a cascading set in which regular expressions are referred to by ids and can embed further regular expressions. For example, the top-level regular expression for matching a ^1H NMR spectrum is as follows;

```
<node type="spectrum" id="hnmr" value="hnmr">
  <regexp parsegroup="0">
    <insert idref="nmrDelta" />?
    \b
    <insert idref="hNmr.Prolog"/>
    (? \W* for\s+\w+ (? (![\(\);])*)?)?
    <insert idref="nmrMethod"/>?
    (\W+(<insert idref="nmrDelta"/>|H)+\b)?
    [\s:=]+?
    (? \W*ppm\W*?)?
    (? peaks\s+at\s+)?
    (? \s*<insert idref="nmrDelta"/>\s+)?
    <insert idref="nmrPeakBlock"/>
  </regexp>
  <child type="quantity" id="hnmrSolvent"/>
  <child type="quantity" id="hnmrStandard"/>
  <child type="quantity" id="hnmrFrequency"/>
  <child type="quantity" id="hnmrTemperature"/>
  <child type="peaks" id="hnmr"/>
</node>
```

Figure 2-8: ^1H NMR regular expression

The `node` element denotes a concept to be annotated, the value of the `type` attribute gives the name of the tag to be assigned to the annotation and the `id` attribute gives the value of the `type` attribute to set on the annotation, as can be seen above in the annotated ^1H NMR – in this case, `<spectrum type="hnmr">`. Other regular expressions are inserted into the top-level regex using

the `insert` elements, and important subsections of the matching text that are also to be annotated are defined using the `child` tag – where the regular expressions defined for these children are matched then the annotations produced by OSCAR3 will have children accordingly. For illustration, note that the frequency and peaks in the annotated spectrum above are children of the spectrum element, as defined by the third and fifth children of the top-level ^1H NMR regular expression given above.

2.2.7 ChemicalTagger

ChemicalTagger (14) is a tool, developed within the Unilever Centre, for the syntactic analysis of chemical documents. By combining the capacity of OSCAR3 for the identification of chemical named entities with common linguistic tools it is possible to identify the grammatical structure of chemical text and then use this information to infer the meaning. This is achieved using the Natural Language Processing techniques of tagging (the process of assigning tags to denote the meaning of individual tokens) and chunking (the process of building the grammatical structure according to a set of production rules). The functioning of ChemicalTagger may be summarised as follows;

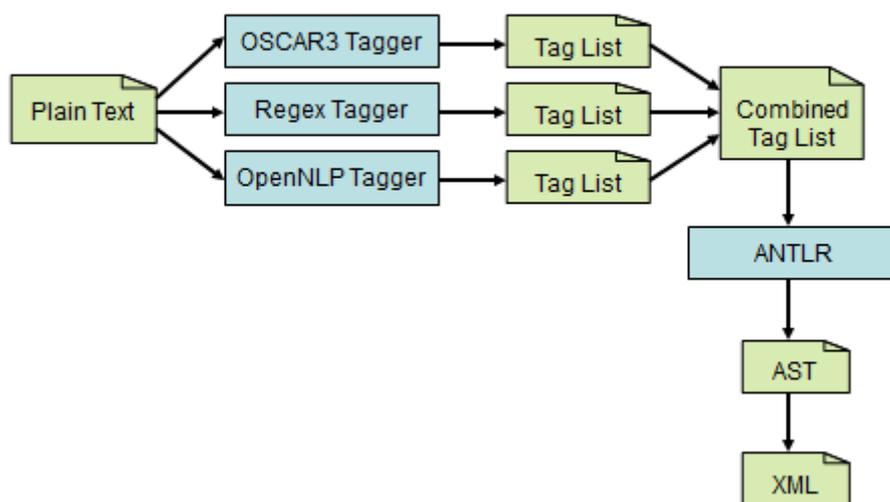


Figure 2-9: ChemicalTagger Architecture

ChemicalTagger accepts as its input format plain text. This text is then passed separately through each of three taggers to create three separate tag sets, which are then combined using heuristics to ascertain priority of the tags from each set. ANTLR (Another Tool for Language Recognition) (58) is then used to parse this single tag set according to a predefined formal grammar to determine its structure in a process known as chunking, and the result is produced as an Abstract Syntax Tree (AST), which is subsequently converted to XML and returned to the user.

2.2.7.1 Tagging

Tagging is the process of selecting an appropriate label to denote the meaning of each token in a sequence. In the ChemicalTagger workflow, the input text is tokenised on whitespace before three separate taggers are applied to create three separate tag sets. The three taggers are employed as they categorise the token based on different criteria, as described below.

OpenNLP Tagger

The `OpenNLPTagger` class passes the input text to the part-of-speech (POS) tagger provided by OpenNLP (59). This implementation is based on a MEMM and is trained with data from the Wall Street Journal and the Brown corpus. The POS tags that are applied to the text describe the function of the word within the sentence, *e.g.* noun or verb. These broad categories are subdivided into specific classes *e.g.* singular proper noun or past participle. The POS tags provide much of the key information that is required in order to identify the grammatical structure of the text. For example, the input string “The cat sat on the mat” is tagged as follows;

The	cat	sat	on	the	mat
DT	NN	VBD	IN	DT	NN

where the POS tags are those from the Penn Treebank tagset (60), and have the following meaning;

- DT – determiner
- NN – noun, singular or mass
- VBD – verb, past tense
- IN – preposition/subordinating conjunction

OSCAR3 Tagger

The `OscarTagger` class passes the text to OSCAR3 to be annotated, using the default parameters. Tokens that are annotated as named entities by OSCAR3 are tagged as being of the type assigned to them by OSCAR3. Tokens that occur as part of a multiple token named entity *e.g.* “ethyl acetate” are further tagged with their position in that named entity, *i.e.* start, middle or end to enable simpler reconstruction later in the `ChemicalTagger` workflow.

Regex Tagger

The `RegexTagger` class applies a series of regular expressions to each token – those that match a regular expression are given the tag associated with that regular expression. For example, the token “mol” and variations thereon are given the tag “NN-AMOUNT” and are matched by the regular expression;

```
(\d\d+)?(m|k)?mol(s)?$
```

By tagging the token “mol” in this way, it is possible to recognise a molar amount in a chemical text as a number followed by a token of type NN-AMOUNT. Such multi-token entities are defined in a set of formal production rules and are recognised during the chunking process.

2.2.7.2 Combination of Tag Sets

The combination of the three separate tag lists into one combined set takes part in two stages. Initially, a simple rule of priority is used – if the token has received a tag from OSCAR3, other than ONT, then that tag takes priority. Otherwise, tags assigned by the regex tagger take priority over those from the OpenNLP tagger. A second step of processing then corrects a number of common errors. For example, the token “M” will be annotated as CM by OSCAR3, being a common abbreviation for “metal”. However, if the preceding token is a number, as in “2 M”, then it is likely that it is being used as abbreviation for “molar”. This error is corrected by reassigning the tag as “NN-MOLAR”. Such error correction serves to allow correct chunking in the next stage of the `ChemicalTagger` workflow.

2.2.7.3 Chunking – Determination of Grammatical Structure

Using a pre-defined regular grammar, ChemicalTagger is able to determine the grammatical structure of a given input text. In this grammar, the format of the expected language is defined using a series of production rules that reference one another and the tags assigned to the input tokens. For example, the quantity of a chemical compound as commonly reported in the literature, such as “0.14 mol, 2.89 g”, is specified as follows;

```
nnameamount: 'NN-AMOUNT' TOKEN;
```

meaning that an “nnameamount” is a token that has been tagged as “NN-AMOUNT” in the combined tag set.

```
amount : cd nnameamount -> ^(NODE["AMOUNT"] cd nnameamount );
```

meaning that an “amount” is composed of a cardinal number (“cd”) followed by an “nnameamount”. The part of the rule following the symbol “->” is an instruction to group the matched content into a single “AMOUNT” entity.

```
measurementtypes : molar|amount|mass|percent|volume ;
```

defining the five types of common measurements

```
measurements : (cd nn)? measurementtypes dt?;  
quantity2 : measurements (comma measurements)* ;  
quantity : (quantity1|quantity2) -> ^(NODE["QUANTITY"] quantity1?  
quantity2?);
```

meaning that a “quantity” may be formed from one or more “measurementtypes” in a comma separated list. Similar sets of rules define grammatical structures such as noun phrases and verb

phrases. As a result, the input text “Into a pressure reactor was placed 4-(dimethylamino)-benzenethiol (.147 g, .96 mmol).” produces the output shown in Figure 2-10, once converted into XML.

```
<Document>
  <PrepPhrase>
    <IN-INTO>Into</IN-INTO>
    <NounPhrase>
      <DT>a</DT>
      <APPARATUS>
        <NN-PRESSURE>pressure</NN-PRESSURE>
        <NN-APPARATUS>reactor</NN-APPARATUS>
      </APPARATUS>
    </NounPhrase>
  </PrepPhrase>
  <VerbPhrase>
    <VBD>was</VBD>
    <VB-SUSPEND>placed</VB-SUSPEND>
  </VerbPhrase>
  <NounPhrase>
    <MOLECULE>
      <OSCAR-CM>4-(dimethylamino)-benzenethiol</OSCAR-CM>
    <QUANTITY>
      <_-LRB-> (</_-LRB->
      <MASS>
        <CD>.147</CD>
        <NN-MASS>g</NN-MASS>
      </MASS>
      <COMMA>,</COMMA>
      <AMOUNT>
        <CD>.96</CD>
        <NN-AMOUNT>mmol</NN-AMOUNT>
      </AMOUNT>
      <_-RRB->)</_-RRB->
    </QUANTITY>
  </MOLECULE>
</NounPhrase>
<STOP>.</STOP>
</Document>
```

Figure 2-10: Sample ChemicalTagger output

Once the meaning of chemical text has been determined and formally encoded in this manner, it is possible for a machine to automatically interpret the meaning of the input text. In the example above, for example, by using XPath it is possible to identify the presence and amount used of 4-(dimethylamino)-benzenethiol in the reaction from which the input text was taken. In this way, ChemicalTagger may be used as the foundation of a system for the semanticizing of texts that report

chemical syntheses. In the current work, ChemicalTagger revision 321256e8b429 (dated 2010-06-10) is used throughout.

2.2.8 OSRA

In spite of the proliferation of machine-understandable chemical formats, much structural information is still communicated in image formats. To the document's author and primary readership, this method is not without its advantages – in many situations, a chemical structure may be communicated to a trained chemist better and faster by using a structure diagram than by a systematic name. Indeed, when describing a chemical reaction the use of a reaction diagram greatly clarifies matters as compared to describing it in text. To a human, the reaction scheme shown in Figure 2-11 is preferable to the phrase “Isopropyl 3-(hydroxymethyl)pyridine-2-carboxylate plus methyl N-[(4-methylphenyl)sulfonyl]glycinate plus diethyl azodicarboxylate plus triphenylphosphine gives 3-[[Methoxycarbonylmethyl-(toluene-4-sulfonyl)-amino]-methyl]-pyridine-2-carboxylic acid isopropyl ester”. The use of image formats to communicate structural information is thus understandable, though in the context of this work unhelpful.

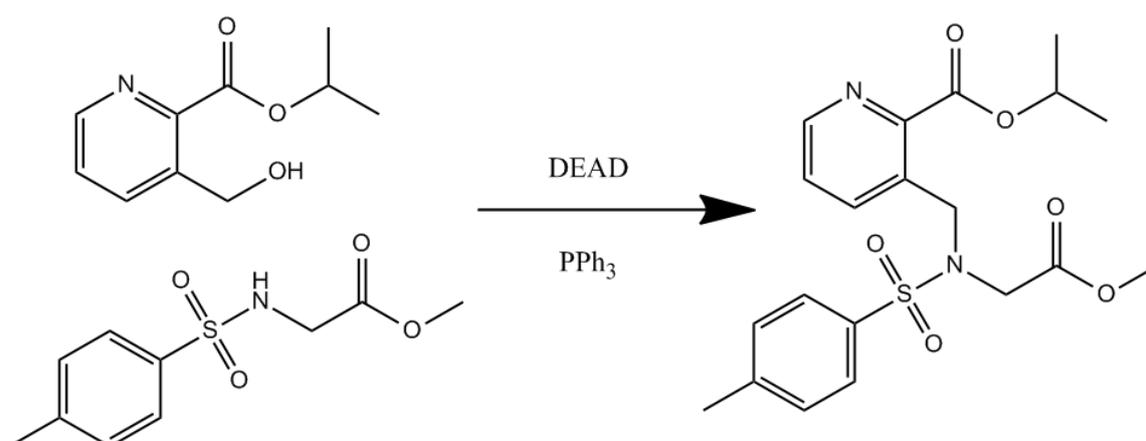


Figure 2-11: Example reaction scheme

The resolution of structure diagrams to connection table is a simple task for a human; however to a machine the images do not directly contain any structural information. The creation of software for this task has been attempted by a number of groups, resulting in a number of applications including Kekulé (61), CLiDE (62; 63; 64), OSRA (65; 66; 67), chemOCR and ChemReader (68). This research dates back to the 1990's, and has in the last few years been undergoing a renaissance – in particular the development of the Optical Structure Recognition Application (OSRA), the first open-source application for the interpretation of chemical structure diagrams. The published systems follow similar methodology in the analysis of input images, employing modules to carry out the following tasks;

- Optical Character Recognition (OCR) – the identification of text within the image, which is typically used to denote atoms other than carbon.
- Vectorisation – the process of reducing the width of lines and transforming the raster (bitmap) image to a vector format.
- Bond detection – the identification of which graphical elements constitute bonds, and which types of bonds are represented *e.g.* wedge bonds, double bonds, and aromatic bonds.
- Connection table compilation – the process of turning the information gathered in the previous steps into a connection table.

Comparison of the currently available applications was performed by Park *et. al.* (68), who concluded that on a corpus of 362 images ChemReader performed best, followed by OSRA, Kekulé and CLiDE. This result should be treated with caution, however, since new versions of both OSRA and CLiDE have been released since the work was performed – and it should be noted that Park *et. al.* are the authors of ChemReader. Indeed, it is only that due to this fact that they could compare the performance of ChemReader since their software is not available externally pending the removal of open-source components that prevent the release of a commercial version. Since OSRA is, at

present, the only open-source solution for chemical image interpretation it was selected for the current work as the use of one of the alternative pieces of software would preclude the release of the software created during the preparation of this thesis as open-source.

2.3 Conclusions

This chapter discussed the current situations of chemical document availability and the leading technologies that have been brought to bear on the problems of information extraction from the literature and the reuse of such information – two areas of critical importance for the current work. The applications of these technologies and the information that can be automatically recovered from the literature are subsequently described in chapters 4 and 5.

3. Representation and Manipulation of Markush Structures

To be effective, a chemical patent must not only claim protection over those compounds that the authors have prepared and demonstrated to be of value, but also prevent competitors from introducing trivial modifications and thereby circumventing the protection that the patent grants. To achieve this goal, chemical patents describe the subjects of the invention in terms of generic structures – known as Markush structures after Eugene Markush, the first inventor to employ them – in which a number of related chemical species are simultaneously described by a shared scaffold with a number of variable features. For example, in a trivial case the set of three monochlorinated toluenes may be described by the generic structure shown in Figure 3-1.

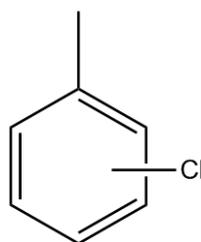


Figure 3-1: Generic structure representing the monochlorinated toluenes

The Markush structures employed in patents are far more complex than that depicted in Figure 3-1, featuring multiple independently variable features to simultaneously describe hundreds of thousands of specific chemical compounds or more that could not feasibly be enumerated within the document. Since the Markush structures describe the area of chemical space for which a patent claims protection, the information they contain is of crucial importance. For that reason, information systems that deal with chemical patents require the capability to represent and handle the Markush structures that the patents contain.

The 1980's saw the development of a number of systems for the representation of Markush structures including the two main commercial systems (69), Markush DARC and MARPAT (70; 71). In

addition, a large project at the University of Sheffield led to the development of the GENSAL language (72; 73) which has since fallen into disuse. The current state of the art is sufficient to allow the creation of Markush database searching tools such as STN, but incompatible with the semantic web of chemistry. Since CML does not permit the description of Markush structures, it was a goal of the current work to explore how the existing technology of Polymer Markup Language (PML) could be employed to create this capability. The work presented in this chapter is therefore intended to serve as a proof of principle rather than as a fully-functional system that is ready for a large-scale implementation.

3.1 Markush Structures

The manner in which Markush structures are used to describe areas of chemical space is consistent across the patent literature. Structural diagrams are used to illustrate the common scaffolds and how the variable features interact with them. Pseudoatoms are used to indicate the presence of complex features, while accompanying text describes the chemical identity of these groups. The classes of variability used in Markush structures are described by Barnard and Downs (74) as follows;

- Substituent variation, where a group represents one unit chosen from an explicitly enumerated list of possibilities, *e.g.* “R1 is fluorine or chlorine”.
- Homology variation, where a group represents one unit chosen from an implied list of possibilities by use of a class name, *e.g.* “R1 is alkyl” or “R1 is a halogen”.
- Frequency variation, where the number of repetitions of a motif is allowed to vary, *e.g.* “R1 is $-(\text{CH}_2)_n\text{OH}$, where n is 0 to 12”.
- Position variation, where the position to which a substituent is bonded is not fixed, *e.g.* the case of the chlorinated toluenes seen in Figure 3-1.

It should be appreciated, however, that these four classes of variation are conceptually rather than fundamentally different from one another since the sets of substituents which match a given example of homology, frequency or position variation are in principle enumerable and could therefore be expressed in terms of substituent variation. Of course, such an enumeration is not always practical – the problem of combinatorial explosion renders the enumeration of long-chain branched alkyl groups unworkable, for example. The usage of the other forms of variation where appropriate is also of value to the creation of an understandable Markush structure if these forms are a true representation of the variation which the author intends to describe.

3.2 Polymer Markup Language

Earlier work in the Unilever Centre for Molecular Science Informatics has resulted in the development of Polymer Markup Language (PML), a CML vocabulary for the fragment-based description of polymers (75). In PML, polymer structures are described as assemblages of `fragment` elements. A `fragment` acts as a container for further `fragment` elements and for `molecule` elements. These `molecule` elements define atomistic information and represent molecular substructures in that one or more of the atoms in the `molecule` is a pseudoatom, intended to denote the position of a free valency. These free valencies are referenced by `join` elements, which describe the molecular connectivity between the substructures. The usage of PML is further discussed with reference to specific examples in sections 3.2.1, 3.2.2 and 3.2.3.

3.2.1 Representation of Polyethylene Oxide

As a simple example of the usage of PML, the representation of a pentamer of polyethylene oxide is shown in Figure 3-2.

```

1 <fragment xmlns:g="http://www.xml-cml.org/mols/geom1">
2   xmlns="http://www.xml-cml.org/schema">
3   <fragment countExpression="*(5)">
4     <join order="1" moleculeRefs2="PREVIOUS NEXT"
5       atomRefs2="r2 r1">
6       <torsion>180</torsion>
7     </join>
8     <fragment>
9       <fragment>
10        <molecule ref="g:o"/>
11      </fragment>
12      <join id="j1" order="1" moleculeRefs2="PREVIOUS NEXT"
13        atomRefs2="r2 r1">
14        <torsion>180</torsion>
15      </join>
16      <fragment>
17        <molecule ref="g:ch2"/>
18      </fragment>
19      <join id="j2" order="1" moleculeRefs2="PREVIOUS NEXT"
20        atomRefs2="r2 r1">
21        <torsion>180</torsion>
22      </join>
23      <fragment>
24        <molecule ref="g:ch2"/>
25      </fragment>
26    </fragment>
27  </fragment>
28 </fragment>

```

Figure 3-2: PML representation of poly(ethylene oxide)

The repeat unit of poly(ethylene oxide) (PEO) is specified by the `fragment` on line 8 of Figure 3-2. This `fragment` contains three further `fragment` elements, on lines 9, 16 and 23. Each of these three `fragment` elements contains a `molecule` element. The atomistic representations (connection tables) for the subunits are contained within separate resources, and are referred to using namespaced identifiers. An example of one of these substructures is given in Figure 3-3.

```

1 <molecule xmlns="http://www.xml-cml.org/schema" id="o">
2   <atomArray>
3     <atom id="a1" elementType="O" x3="-0.000000"
4       y3="-50.065000" z3="0.000000"/>
5     <atom id="r1" elementType="R" x3="-0.776000"
6       y3="0.512000" z3="-0.000000"/>
7     <atom id="r2" elementType="R" x3="0.776000" y3="0.512000"
8       z3="-0.000000"/>
9   </atomArray>
10  <bondArray>
11    <bond atomRefs2="a1 r1" order="1"/>
12    <bond atomRefs2="a1 r2" order="1"/>
13  </bondArray>
14 </molecule>

```

Figure 3-3: Atomistic representation for the g:o fragment

This molecule contains three atoms – an oxygen atom which is bonded to two pseudoatoms of elementType R. Similarly the other fragment used in the description of polyethylene oxide, the methylene fragment, may be found to consist of a -CH₂- unit, the carbon atom of which is connected to two of these pseudoatoms. The role of these pseudoatoms, referred to as R-groups, is to indicate the free valencies present in a fragment through which joins may be used to connect it to further fragments and thereby to construct a macromolecule.

The three `molecule` elements used in the PML document shown in Figure 3-2 are connected by the two `join` elements on lines 12 and 19. Each of these joins specifies a bond order, as well as a `moleculeRefs2` attribute and an `atomRefs2` attribute. The value of `moleculeRefs2` used here, `PREVIOUS NEXT`, indicates that the fragments should be connected sequentially as opposed to as a side-chain. The `atomRefs2` attribute indicates which of the pseudoatoms in the individual fragments should be used in the construction of the bond. The value used in this example, `r2 r1`, indicates that the dummy atoms to use are the one with an id of `r2` in the `PREVIOUS` fragment and the one with an id of `r1` in the `NEXT` fragment. These dummy atoms are to be deleted and new bonds between the fragments created as shown in Figure 3-4. The `join` elements may contain a child `torsion` element, which defines an appropriate torsion angle for the new bond.

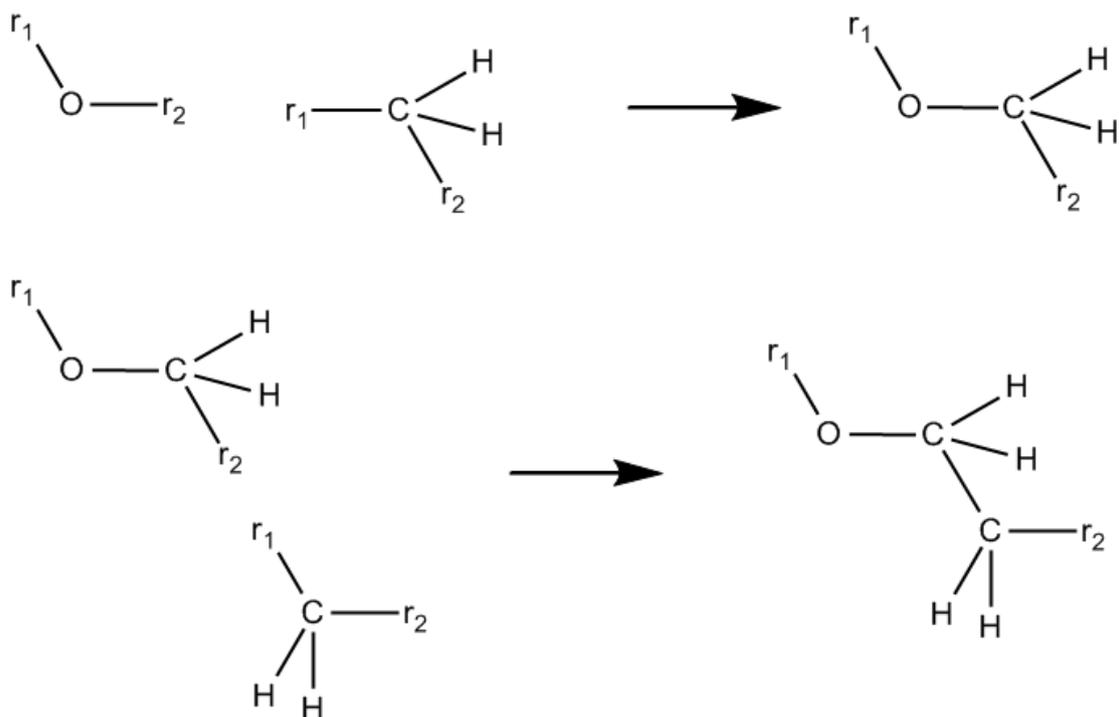


Figure 3-4: The creation of bonds between fragments in PML

The `fragment` on line 8 is itself contained by the `fragment` on line 3. The `fragment` in this position – a child of the root element – defines the constitutional repeating unit of the polymer and is hereafter referred to as the *primary fragment*. The optional `countExpression` attribute on the primary fragment specifies the number of repeat units present, in this case five, while the `join` on line 4 specifies how the repeat units are connected to one another using the syntax covered previously.

3.2.2 Representation of Polystyrene

The example of polyethylene oxide discussed above is simple in that the macromolecule may be described as a continuous chain with no side chains. This topology is by no means ubiquitous, and consequently PML supports the description of branched polymers such as comb polymers and dendrimers. The methods which permit the description of branched polymers may also be employed

to describe polymers with a single backbone but which also carry side chains. This phenomenon is illustrated in Figure 3-5.

```
1 <fragment xmlns:g="http://www.xml-cml.org/mols/geom1"
2   xmlns="http://www.xml-cml.org/schema">
3   <fragment countExpression="*(5)">
4     <join order="1" moleculeRefs2="PREVIOUS NEXT"
5       atomRefs2="r2 r1">
6     </join>
7     <fragment>
8       <fragment>
9         <molecule ref="g:ch">
10          <join order="1" moleculeRefs2="PARENT CHILD"
11            atomRefs2="r3 r1">
12            <fragment>
13              <molecule ref="g:benzene"/>
14            </fragment>
15          </join>
16        </molecule>
17      </fragment>
18    <join order="1" moleculeRefs2="PREVIOUS NEXT"
19      atomRefs2="r2 r1">
20    </join>
21    <fragment>
22      <molecule ref="g:ch2"/>
23    </fragment>
24  </fragment>
25 </fragment>
26 </fragment>
```

Figure 3-5: PML representation of polystyrene

It can be seen that the usage of the `join` element on line 10 of the PML document in Figure 3-5 differs from that seen previously. Here, the `moleculeRefs2` attribute takes the value `PARENT CHILD`, which indicates the presence of a side chain. The `fragment` on line 12 that contains the molecular subunit corresponding to the side chain is present as a child of the `join` element that connects it to the polymer backbone, while the `join` element is itself a child of the `molecule` on line 9 where previously the three elements would have been siblings. This format provides a convenient means to represent branching in a macromolecule, as it permits an arbitrary number of branches from the same position, and the branching in the XML document directly matches that in the macromolecule described, creating a comprehensible document.

3.2.3 Representing Variability in PML

The examples of polyethylene oxide and polystyrene illustrate how PML may be used to represent a macromolecule with a precisely defined structure; however, it is rare for the composition of a polymer to be precisely known. Polymerisation processes can be difficult to control and resultant samples of polymers are polydisperse, *i.e.* composed of macromolecules of varying chain length, while specific mechanisms of polymerisation may lead to variable degrees of branching or for asymmetric monomers such as propylene to be incorporated in a head-to-tail fashion. This variability of polymer structure is further demonstrated by the cases of random and statistical copolymers, in which the positioning of the monomeric units is determined by a stochastic process.

In order to fully support these more complex cases, PML must permit these forms of variability to be fully described. This support is provided by describing some of the `fragment` elements in the document as lists of fragments from which one is to be selected. The usage is demonstrated in Figure 3-6, which describes a PML representation of a statistical copolymer of ethylene oxide and propylene oxide.

```
1 <fragment xmlns='http://www.xml-cml.org/schema'  
2   xmlns:g='http://www.xml-cml.org/mols/geom1'>  
3   <fragmentList>  
4     <fragment id='eo'>  
5       <molecule ref='g:ethyleneOxide' />  
6     </fragment>  
7     <fragment id='po'>  
8       <molecule ref='g:propyleneOxide' />  
9     </fragment>  
10    <fragment id='eE'>  
11      <fragment ref='eo' />  
12      <join atomRefs2='r2 r1' moleculeRefs2='PREVIOUS NEXT' />  
13      <fragment ref='EE' />  
14    </fragment>  
15    <fragment id='eP'>  
16      <fragment ref='eo' />  
17      <join atomRefs2='r2 r1' moleculeRefs2='PREVIOUS NEXT' />  
18      <fragment ref='PP' />  
19    </fragment>  
20    <fragment id='EE'>  
21      <fragmentList role='markushMixture'>  
22        <fragment ref='eo'>
```

```

23         <scalar dictRef='cml:ratio' dataType='xsd:double'>
24             0.01
25         </scalar>
26     </fragment>
27     <fragment ref='eE'>
28         <scalar dictRef='cml:ratio' dataType='xsd:double'>
29             0.84
30         </scalar>
31     </fragment>
32     <fragment ref='eP'>
33         <scalar dictRef='cml:ratio' dataType='xsd:double'>
34             0.15
35         </scalar>
36     </fragment>
37 </fragmentList>
38 </fragment>
39 <fragment id='pE'>
40     <fragment ref='po' />
41     <join atomRefs2='r2 r1' moleculeRefs2='PREVIOUS NEXT' />
42     <fragment ref='EE' />
43 </fragment>
44 <fragment id='pP'>
45     <fragment ref='po' />
46     <join atomRefs2='r2 r1' moleculeRefs2='PREVIOUS NEXT' />
47     <fragment ref='PP' />
48 </fragment>
49 <fragment id='PP'>
50     <fragmentList role='markushMixture'>
51         <fragment ref='po'>
52             <scalar dictRef='cml:ratio' dataType='xsd:double'>
53                 0.01
54             </scalar>
55         </fragment>
56         <fragment ref='pP'>
57             <scalar dictRef='cml:ratio' dataType='xsd:double'>
58                 0.84
59             </scalar>
60         </fragment>
61         <fragment ref='pE'>
62             <scalar dictRef='cml:ratio' dataType='xsd:double'>
63                 0.15
64             </scalar>
65         </fragment>
66     </fragmentList>
67 </fragmentList>
68 </fragment>
69 <fragment id='f0'>
70     <fragment ref='EE' />
71 </fragment>
72 </fragment>

```

Figure 3-6: PML representation of a statistical copolymer

In the PML document shown in Figure 3-6, a number of the fragments are defined by means of reference, such as that on line 70. The content of these fragments, such as the monomeric units

from which the polymer is constructed on lines 4 and 7, is then defined in the *primary fragment list* which is a child of the root element and is defined in the PML document above on line 3. The `fragment` elements that define the monomeric units are used in the construction of the `eE`, `eP`, `pE` and `pP` fragments on lines 10, 15, 39 and 44 respectively, each of which consists of an ethylene oxide (`e`) or propylene oxide (`p`) unit followed by a list from which to draw the next unit (`EE` or `PP`, defined on lines 20 and 49 respectively). These lists, called “Markush mixtures”, specify that the next unit to be used is either one of the `eE`, `eP`, `pE` and `pP` fragments, representing a continuation of the polymer chain, or a single polyethylene or polypropylene oxide unit, representing a termination condition. The probabilities with which each of the possible fragments is selected are specified in the `scalar` elements that are children of the `fragment` concerned, such as that on line 23. The primary fragment is defined on line 69 and points to a single `EE` element, indicating that the chains start with an ethylene oxide unit though this could just as easily have been replaced by a further Markush mixture allowing the initial unit to be either ethylene or propylene oxide.

The capacity afforded by the `markushMixture` to represent a list of potential fragments allows for a variety of variability exhibited by polymers to be described in PML. The head-to-head or tail-to-tail addition of propylene monomers, for example, could have been described in the example above by the definition of the `pO` fragment as a `markushMixture` list consisting of a head-first and a tail-first propylene unit while branched polymers may be described by the inclusion of a branching point in a `markushMixture` list that specifies the repeating units of the backbone of a polymer. This flexibility is key to the representation of complex sets of macromolecules and provides a means to describe the variability exhibited by Markush structures, as shall be discussed in section 3.3.

3.2.4 The Cambridge Polymer Builder

In order to demonstrate the usage of PML, a demo application was developed (76). The Cambridge Polymer Builder is a frontend application that permits a user to create atomistic CML macromolecules with 3D-coordinates from a restricted set of pre-generated fragments. The application runs as a web service and is accessed using a web browser. The front page of the Cambridge Polymer Builder is shown in Figure 3-7.

The screenshot shows a web browser window titled "Polymer Builder - Mozilla Firefox" with the URL <http://wwwmm-svc.ch.cam.ac.uk/polydemo/>. The page content includes:

- Polymer Builder** header.
- Templates** section with five options:
 - Fixed Length Linear Homopolymer**: R_1 — [red bar] — R_2 AN
 - Fixed Length Block Copolymer**: R_1 — [red bar] — [grey bar] — R_2 $ANBM$
 - Alternating Copolymer**: R_1 — [red bar] — [grey bar] — [red bar] — [grey bar] — R_2 $(AN_xBN_y)_M$
 - Random Copolymer**: R_1 — [red bar] — [grey bar] — [red bar] — [grey bar] — R_2 $A\text{-ran-}B$
 - Fixed Length Comb Polymer**: R_1 — [red bar with diagonal lines] — R_2 $(A(BN_x))_NA$
- Dendrimer**: A diagram of a red spherical dendritic structure.
- Unilever Cambridge** logo and "Centre For Molecular Science Informatics" text.
- A 3D ball-and-stick molecular model of a polymer chain.
- Text "Powered by JUMBO" with a small elephant icon.

Figure 3-7: The front page of the Cambridge Polymer Builder

The front page of the application prompts the user to select a polymer topology from the list of linear homopolymer, block copolymer, alternating copolymer, random copolymer, comb polymer and dendrimer. When the user selects one of these topologies, he is presented with a form (such as

that illustrated in Figure 3-8) into which he enters the required parameters, *i.e.* the identities of the repeat units and end groups to be used, the degrees of polymerisation and the torsion angles to be used around the bonds formed between repeat units.

Fixed length linear homopolymer

R1 XXXXXXXXXX R2

Degree of polymerisation

Torsion angle

End group 1	Repeat Unit	End group 2
<input type="radio"/> Acetoxy	<input checked="" type="radio"/> Acrylic Acid	<input type="radio"/> Acetoxy
<input type="radio"/> Acetyl	<input type="radio"/> cis-Acetylene	<input type="radio"/> Acetyl
<input type="radio"/> Bromane	<input type="radio"/> trans-Acetylene	<input type="radio"/> Bromane
<input type="radio"/> Chlorane	<input type="radio"/> Acylonitrile	<input type="radio"/> Chlorane
<input type="radio"/> Ethyl	<input type="radio"/> para-Benzene	<input type="radio"/> Ethyl
<input type="radio"/> Fluorine	<input type="radio"/> Difluoroethylene	<input type="radio"/> Fluorine
<input type="radio"/> Hydrogen	<input type="radio"/> Dimethylbutene	<input type="radio"/> Hydrogen
<input type="radio"/> Hydroxy	<input type="radio"/> Dimethylsiloxane	<input type="radio"/> Hydroxy
<input type="radio"/> Methoxycarbonyl	<input type="radio"/> Ethylene	<input type="radio"/> Methoxycarbonyl
<input type="radio"/> Methyl	<input type="radio"/> Ethylene Oxide	<input type="radio"/> Methyl
<input type="radio"/> Propanoic Acid	<input type="radio"/> Ethylene Glycol	<input type="radio"/> Propanoic Acid
	<input type="radio"/> α -D-glucose	
	<input type="radio"/> Glycine	
	<input type="radio"/> L-Alanine	
	<input type="radio"/> Methyl Methacrylate	
	<input type="radio"/> Naphthalene	
	<input type="radio"/> Propylene Oxide	
	<input type="radio"/> Propylene	
	<input type="radio"/> Propylene Glycol	
	<input type="radio"/> Styrene	
	<input type="radio"/> Triazene	
	<input type="radio"/> Vinylalcohol	
	<input type="radio"/> Vinylchloride	

[Build another polymer](#)

Unilever Cambridge
Centre For Molecular Science Informatics

Powered by JUMBO

Figure 3-8: Designing a polymer

The parameters selected by the user are used to populate a template PML document for the selected topology. The FragmentTool class from the JUMBO library is then used to construct either the atomistic CML macromolecule that is described by the PML document or an atomistic CML macromolecule that corresponds to the description given in the PML document if the input document defines a variable polymer. This process requires dereferencing the `molecule` elements that define the structural subunits of the polymer and combining them together in the manner

described in the input document. As the polymer chain is built, each structural fragment is automatically rotated to align it with the chain and to set the torsion angle about the new bond equal to that specified by the user, and positioned such that the atoms from either fragment are a distance apart equal to the sum of their covalent radii. Where appropriate, such as in the building of a random copolymer, individual structural subunits are selected at random from a markushMixture to create a single macromolecule that exemplifies the polymer described in the PML document. In addition, where possible, molecular properties are calculated using the van Krevelen group contribution method (77). The user is then presented with a page depicting these results, such as that shown in Figure 3-9, and from which the CML description of the macromolecule can be downloaded.

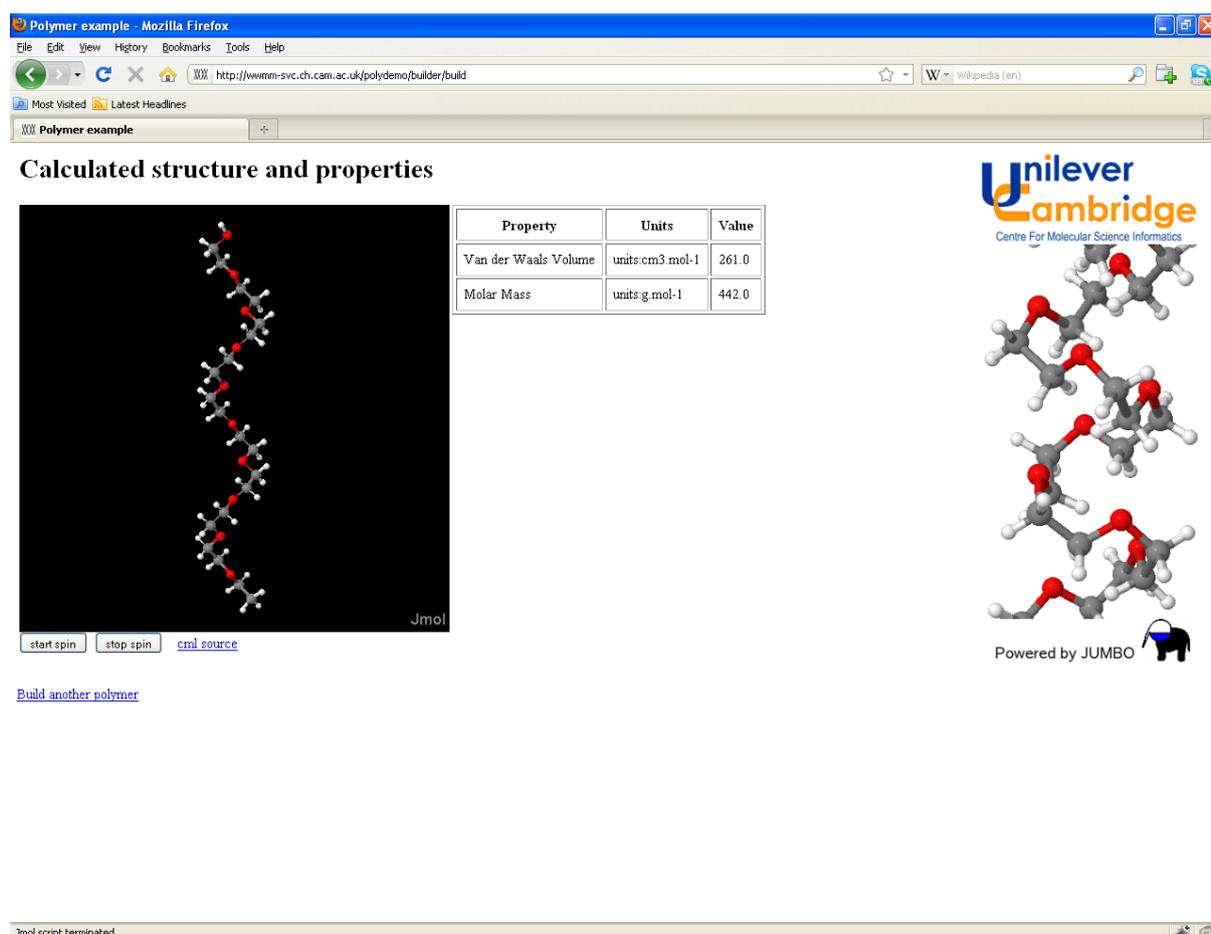


Figure 3-9: Results of polymer building

3.3 Extension of PML for Markush Structures

When commencing this work, it was thought that Polymer Markup Language would provide a suitable basis from which to develop a CML-based method for Markush structure representation. The capacity, using PML, to represent molecular substructures by the `fragment` element and to describe, using a `markushMixture`, a list of the allowed substituents at a given position provides at a basic level the functionality required to describe Markush structures. As discussed previously, Barnard's four classes of variability within Markush structures may be reduced to substituent variation, which PML is equipped to handle, if the user is willing to fully enumerate the set of substituents that a patent author claims for a given position. This strategy, however, can lead to extremely verbose descriptions that are unlikely to be acceptable to a user. The number of mono-substituted alkane derivatives, and therefore also alkyl radicals, containing n carbon atoms exceeds 500 for $n = 10$ and 5,000,000 for $n = 20$ (78), while full enumeration of position variation in PML obfuscates the author's intention. Consider, for example, the case of the monochlorinated toluenes shown in Figure 3-1; this simple Markush structure could be represented in PML as shown in Figure 3-10, in which the substituted benzene ring is defined three times – once each for the *ortho*-, *meta*- and *para*-substituted motif.

```
<fragment xmlns='http://www.xml-cml.org/schema'
  xmlns:g='http://www.xml-cml.org/mols/geom1'>
  <fragment id='f0'>
    <fragment ref='methyl' />
    <join atomRefs2='r1 r1' moleculeRefs2='PREVIOUS NEXT' />
    <fragment ref='chlorinatedBenzene' />
  </fragment>

  <fragmentList>
    <fragment id='methyl'>
      <molecule ref='g:ch3' />
    </fragment>
    <fragment id='chlorinatedBenzene'>
      <fragmentList role='markushMixture'>
        <fragment ref='orthoChlorinatedBenzene'>
          <scalar dictRef='cml:ratio' dataType='xsd:double'>
            0.4
          </scalar>
        </fragment>
      </fragmentList>
    </fragment>
  </fragmentList>
</fragment>
```

```

    </fragment>
    <fragment ref='metaChlorinatedBenzene'>
      <scalar dictRef='cml:ratio' dataType='xsd:double'>
        0.4
      </scalar>
    </fragment>
    <fragment ref='paraChlorinatedBenzene'>
      <scalar dictRef='cml:ratio' dataType='xsd:double'>
        0.2
      </scalar>
    </fragment>
  </fragmentList>
</fragment>
<fragment id='orthoChlorinatedBenzene'>
  <fragment>
    <molecule ref='g:benzene' />
  </fragment>
  <join order='1' moleculeRefs2='PREVIOUS NEXT'
    atomRefs2='r2 r1'>
  </join>
  <fragment>
    <molecule ref='g:cl' />
  </fragment>
</fragment>
<fragment id='metaChlorinatedBenzene'>
  <fragment>
    <molecule ref='g:benzene' />
  </fragment>
  <join order='1' moleculeRefs2='PREVIOUS NEXT'
    atomRefs2='r3 r1'>
  </join>
  <fragment>
    <molecule ref='g:cl' />
  </fragment>
</fragment>
<fragment id='paraChlorinatedBenzene'>
  <fragment>
    <molecule ref='g:benzene' />
  </fragment>
  <join order='1' moleculeRefs2='PREVIOUS NEXT'
    atomRefs2='r4 r1'>
  </join>
  <fragment>
    <molecule ref='g:cl' />
  </fragment>
</fragment>
</fragmentList>
</fragment>

```

Figure 3-10: PML representation of the monochlorinated toluenes

Instead of mandating full enumeration of acceptable substituents, it is preferable to introduce novel features into the language that permit a more natural and concise expression of the variability in a Markush structure. The remainder of this chapter discusses the solutions that were devised to assist

in the description of Markush structures and that make up Extended Polymer Markup Language – EPML.

3.3.1 Frequency Variation

It has already been discussed how PML may be used to describe a specific number of repeat units using the `countExpression` attribute, *e.g.*

```
<fragment countExpression="*(5)">
  <join order="1" moleculeRefs2="PREVIOUS NEXT"
        atomRefs2="r2 r1">
    </join>
  <fragment>
    ...
  </fragment>
</fragment>
```

In EPML, the `countExpression` is allowed to specify a range of permitted values using the format;

```
<fragment countExpression="range(2,5)">
```

This usage specifies that the fragment in question is permitted to be repeated any number of times within the specified (inclusive) range, thereby describing frequency variation.

3.3.2 Homology Variation

The phenomenon of homology variation, defined earlier as where “a group represents one unit chosen from an implied list of possibilities by use of a class name, *e.g.* ‘R1 is alkyl’ or ‘R1 is a halogen’” may be viewed as a combination of two separate forms of variation – one in which the substituent is enumerable as a precise and well-understood list such as “R1 is a halogen”, and those in which the substituent is defined in terms of a precise or generic structural feature such as “R1 is

an alkoxy group” or “R1 is a substituted or unsubstituted heteroaryl ring”. The description of such features is illustrated in Figure 3-11.

```
1 <fragment xmlns='http://www.xml-cml.org/schema'  
2     xmlns:g='http://www.xml-cml.org/mols/geom1'>  
3   <fragment id='f0'>  
4     <molecule ref='g:benzene'>  
5       <join atomRefs2='r1 r1' moleculeRefs2='PARENT CHILD'>  
6         <fragment homology='halogen' />  
7       </join>  
8       <join atomRefs2='r2 r1' moleculeRefs2='PARENT CHILD'>  
9         <fragment template='alkoxy' branched='true'  
10          minC='1' maxC='4' />  
11       </join>  
12     </molecule>  
13   </fragment>  
14 </fragment>
```

Figure 3-11: Homology variation in EPML

This example shows a benzene ring (specified on line 4) substituted by a halogen (line 6) and in the *ortho*- position by an alkoxy group (lines 9-10) – the R-groups of the g:benzene fragment being numbered consecutively and contiguously around the ring from 1 to 6. The halogen and alkoxy substituents are specified using the `homology` and `template` attributes respectively, the values of which define the ids of the list of permitted substituents and CML molecule against which they should be resolved respectively. The alkoxy group is further specified using the `branched`, `minC` and `maxC` attributes which respectively define whether or not the carbon chain of the alkoxy group is permitted to be branched and the minimum and maximum number of carbon atoms in the alkoxy group, providing support for commonly-used restrictions. The CML molecule to which the alkoxy template resolves is shown in Figure 3-12.

```

1  <molecule id='alkoxy' xmlns='http://www.xml-cml.org/schema'>
2    <atomArray>
3      <atom id='r1' elementType='R' />
4      <atom id='a1' elementType='O' />
5      <atom id='r2' elementType='Q' />
6    </atomArray>
7    <bondArray>
8      <bond id="r1_a1" atomRefs2="r1 a1" order="1" />
9      <bond id="a1_r2" atomRefs2="a1 r2" order="1" />
10   </bondArray>
11 </molecule>

```

Figure 3-12: Formal description of the alkoxy template

The CML molecule that defines the alkoxy template contains three atoms – an oxygen atom on line 4, a pseudoatom of type ‘R’ on line 3 that acts as a free valency by which the substituent may be connected to a parent structure and a pseudoatom of type ‘Q’ on line 5. This new pseudoatom indicates the presence of a carbon chain of variable length. The usage of such pseudoatoms allows the description of such generic substituents as alkyl, alkenyl and cycloalkyl groups, while coverage for more complex structure-based homology variation such as “R1 is a substituted or unsubstituted heteroaryl ring” is not provided. Support for such terminology in state of the art commercial systems is patchy, with Markush DARC allowing the usage of 22 specific “superatoms”, each representing a specific class of substituent such as “aromatic carbocyclic system” and “fused heterocycle”, while MARPAT adopts a similar system of “generic groups” (79). The correct handling of such terminology is a highly complex problem, for which Welford *et al.* (80) propose the usage of a generative grammar to produce a system capable of recognising, for example, a 2-chloro-pyridin-3-yl substituent as being a substituted heteroaryl ring, though this work appears not to have been fully developed. Consequently, there is no available means to fully describe homology variation as it is used by patent authors, and the facilities provided by EPML represent an acceptable approach for such an experimental language.

3.3.3 Position Variation

Position variation, defined earlier as where “the position to which a substituent is bonded is not fixed” is represented formally in EPML in much the same way as it is represented graphically. The mobile substituent is represented in the same as it would be were it to have a fixed position in the structure, while the `join` that attaches it to the main structure has the connectivity information defined in a novel manner as illustrated in Figure 3-13, a representation of the set of monochlorinated toluenes.

```
1 <fragment xmlns='http://www.xml-cml.org/schema'  
2   xmlns:g='http://www.xml-cml.org/mols/geom1'>  
3   <fragment id='f0'>  
4     <molecule ref='g:benzene'>  
5       <join atomRefs2='r1 r1' moleculeRefs2='PARENT CHILD'>  
6         <molecule ref='g:me' />  
7       </join>  
8       <join atomRef1Array='r2 r3 r4' atomRef2Array='r1'  
9         moleculeRefs2='PARENT CHILD'>  
10        <molecule ref='g:cl' />  
11      </join>  
12    </molecule>  
13  </fragment>  
14 </fragment>
```

Figure 3-13: Position variation in EPML

As can be seen above, the `join` that connects the mobile substituent, specified on lines 8-9, does not carry the `atomRefs2` attribute that would normally describe the connectivity between the two fragments. Instead, this information is specified in the `atomRef1Array` and the `atomRef2Array` attributes, which list the ids of the free valencies the `join` is permitted to occupy on the PREVIOUS/PARENT and NEXT/CHILD fragment respectively. If either of these attributes is missing, then it is assumed that the substituent is permitted to be attached to any of the free valencies on the appropriate fragment. For example, if the `atomRef1Array` were missing from the `join` on line 5 of the EPML document in Figure 3-13, it would be assumed that the chlorine group could be

attached to any of the free valencies other than r1, which is occupied by the methyl substituent. This approach to position variation produces EPML documents that are far more concise than their explicitly-enumerated PML counterparts, which may be seen by comparing Figure 3-13 to the PML representation of the same Markush structure given in Figure 3-10.

3.3.4 Position and Count Variation

In addition to position variation, Markush structures sometimes employ position variation that is repeated a variable number of times, as illustrated in Figure 3-14. This Markush structure is subsequently described in EPML in Figure 3-15.

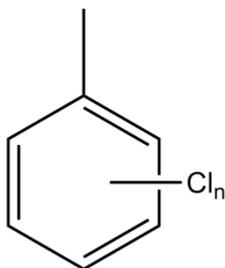


Figure 3-14: Markush structure employing simultaneous position and count variation

```

1 <fragment xmlns='http://www.xml-cml.org/schema'
2   xmlns:g='http://www.xml-cml.org/mols/geom1'>
3   <fragment id='f0'>
4     <molecule ref='g:me' />
5     <join atomRefs2='r1 r1' moleculeRefs2='PREVIOUS NEXT' />
6     <fragment ref='benzene'>
7       <join countExpression='range(2,5)' atomRef2Array='r1'
8         moleculeRefs2='PARENT CHILD'>
9         <fragment>
10          <molecule ref='g:cl' />
11        </fragment>
12      </join>
13    </fragment>
14  </fragment>
15  <fragmentList>
16    <fragment id='benzene'>
17      <molecule ref='g:benzene' />
18    </fragment>
19  </fragmentList>
20 </fragment>

```

Figure 3-15: Simultaneous position and count variation in EPML

The EPML description of the polychlorinated toluene above differs from that of the monochlorinated toluene in Figure 3-13 in that the chlorine substituent is contained within a `fragment` element, on line 9, which is in turn contained within a `join`, on line 7, that carries a `countExpression` attribute. The value of this attribute in the example above specifies a range of between 2 and 5 – specifying the value of the variable n from Figure 3-14, *i.e.* the number of times the substitution unit is repeated.

3.3.5 Inline Connection Tables

The additional features of EPML as compared to PML assist a user in producing concise definitions of Markush structures, but the usage of structural units that are defined in separate documents is a slow process if the units have not previously been defined since a document author must then create atomistic CML representations of the missing units. In order to provide a user with a means to work around this problem, EPML permits the usage of inline connection tables to define the

molecular substructures represented by `fragment` elements. The format in which these connection tables are specified is derived from SMILES, and differs from pure SMILES in two regards. Firstly, the free valencies of the fragment are specified as though they were atoms using the codes “R1”, “R2”, etc. Secondly, since PML lacks a means by which to cyclise a structure the inline connection tables are used as a means to incorporate variability in a cyclised unit. Two forms of variation may be described by an inline connection table – substituent variation and frequency variation. The usage of these features is illustrated in Figure 3-17, which describes the Markush structure shown in Figure 3-16.

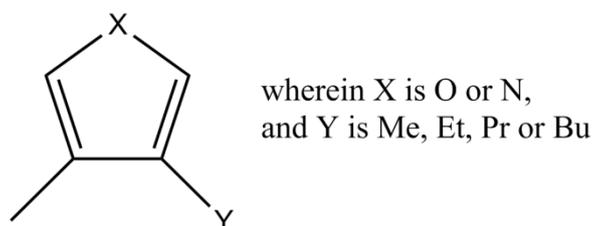


Figure 3-16: Markush structure featuring variable cyclic unit

```

1  <fragment xmlns='http://www.xml-cml.org/schema'>
2    <fragment id='f0'>
3      <fragment smiles='R1C' />
4      <join atomRefs2='r1 r2' moleculeRefs2='PREVIOUS NEXT' />
5      <fragment smiles='R1c1c{O|N}cc1R2' />
6      <join atomRefs2='r1 r2' moleculeRefs2='PREVIOUS NEXT' />
7      <fragment smiles='R1[C[1-4]]' />
8    </fragment>
9  </fragment>

```

Figure 3-17: Inline connection tables in EPML

The `fragment` on line 3 of Figure 3-17, specified by the string “R1C”, represents a methyl group since “C” represents a carbon atom, “R1” represents a free valency and hydrogen atoms are assumed to fill unspecified positions in the SMILES language. In the `fragment` on line 5, substituent variation is indicated by separating the permitted groups with the pipe character (“|”) and wrapping the list in braces (“{” and “}”). Thus, the substring “{O|N}” indicates the presence of either an oxygen or a nitrogen atom. The full string, “R1c1c{O|N}cc1R2”, therefore represents a 3,4-disubstituted

pyrrole or furan, *i.e.* the ring from Figure 3-16. Finally, the fragment on line 7 uses frequency variation to define a carbon chain of between one and four units with the inline connection table “[C[1-4]]”, in which the inner square brackets define the permitted range of integers for the frequency variation, while the text before the inner square brackets define the connection table for the repeated unit – in this case methylene, defined by the string “C”. While the usage of frequency variation in an inline connection table in this example could equally have been replaced by a standard markushMixture, the functionality is useful when describing rings of variable size as an alternative to enumeration.

3.4 Building Representative Examples of a Markush Structure

As discussed earlier, the `FragmentTool` provides the functionality to construct atomistic CML representations of macromolecules described by a PML document. It was desired to create a similar demonstration application for EPML, and it was decided that rather than attempt to re-implement much of the functionality of the `FragmentTool`, this application should instead operate by reducing an EPML document to a PML document which describes a single chemical structure, and from which may be produced an atomistic representation using the `FragmentTool`. This process requires the removal of all of the additional features introduced into EPML and described in section 3.3, and is carried out in a number of steps as follows;

1. Those `fragment` elements that carry a `homology` attribute, *i.e.* that describe homology variation, are selected using XPath. For each such instance, the corresponding `moleculeList` is looked up and one of the substituents from this list is selected at random. The connection table for this substituent is copied from the `moleculeList` and the working document is modified by removing the `homology` attribute and inserting a reference to the selected substituent.

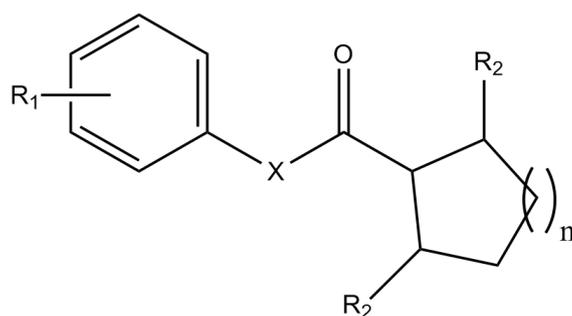
2. Instances of fragments that use inline connection tables are identified by XPath. Those that employ either substituent or frequency variation are selected from this list and the variation defined within each of the inline connection tables is fully enumerated to give the full set of inline connection tables represented by the original. The enumerated list is then used to create a markushMixture of the fragments represented by the set of enumerated inline connection tables, and this markushMixture replaces the original fragment in the working document.
3. Those `join` elements which carry `countExpression` attributes, *i.e.* those used for multiple position variation as discussed in section 3.3.4, are selected by XPath. The `countExpression` attribute is removed from the `join`, which is subsequently copied in position a number of times corresponding to that specified by the `countExpression` attribute. If the value of this attribute specified a range, the number of times to duplicate the `join` is selected at random from those integers specified by the range.
4. Those `fragment` elements which carry `template` attributes are identified using XPath. The connection tables for the templates to which they refer are dereferenced and used to create an inline connection table specifying a single substructure representative of the restrictions carried by the `fragment` in question, *i.e.* that matches the values of the `maxC`, `minC` and `branched` attributes. The `template` attribute is then replaced with this inline connection table in the working document.
5. Those `fragment` elements employing inline connection tables, including those that have been generated during earlier stages of processing, are selected by XPath. For each of these fragments, the inline connection table is converted to a corresponding atomistic CML molecule. Since the `FragmentTool` does not generate 3D co-ordinates, instead requiring them to be provided as an input, they are generated and added to the CML. The inline connection tables are then replaced with references to the newly-created molecules.

The current implementation generates 3D co-ordinates using the CORINA software (81). Since CORINA is commercial software, interfacing with the software is achieved by connecting to a computer in the Unilever Centre on which it is installed. This method does not scale with the number of machines running the MarkushBuilder but is sufficient for demonstration purposes. A preferred method would involve using a distributable open-source solution such as the CDK (82; 83; 84) or OpenBabel but has not been implemented as part of the current work.

6. The markushMixture elements in the working document are selected using XPath. From each of the markushMixtures, a single `fragment` is selected at random. The markushMixture is then replaced in the working document with this fragment. Though this step is not necessary to produce a PML document that can be processed by the `FragmentTool`, it is necessary in order to produce a PML document that defines one and only one structure.
7. The working document is searched by iterative descent for `join` elements. If a `join` does not carry an `atomRefs2` attribute, *i.e.* if the `join` represents position variation, then it is processed to assign a specific `atomRefs2`. Where the `join` carries an `atomRef1Array` or an `atomRef2Array` attribute, *i.e.* where the allowed attachment points on the PREVIOUS/PARENT and NEXT/CHILD fragment respectively have been explicitly stated, then one is selected at random from the list. Where one or both of these lists is missing, a suitable attachment point is determined by examining the fragment concerned, determining a full list of the free valencies on that fragment and removing from this list those valencies that have previously been used by another `join`. Once this process has been carried out, one of the attachment points is selected at random and the `join` undergoing processing is assigned a specific `atomRefs2` attribute.

Once these stages of processing have been carried out, the input EPML document has been reduced to a PML document that describes a single, specific chemical structure. This PML document is then passed to the FragmentTool in order to generate an atomistic CML representation of the specified structure.

The results of this process are illustrated below. A simple Markush structure is described in Figure 3-18 and specified in EPML in Figure 3-19. By applying the MarkushBuilder to this document, an atomistic CML molecule with 3D co-ordinates was produced, which is visualised in Jmol (85) and illustrated as a 2D structure in Figure 3-20.



wherein X is O or N

n is 1-4

R₁ is a halogen

R₂ is C₁-C₄ alkoxy

Figure 3-18: Example Markush structure

```

1 <fragment xmlns="http://www.xml-cml.org/schema"
2   xmlns:g="http://www.xml-cml.org/mols/geom1">
3   <fragment id='f0'>
4     <fragment ref='benzene'>
5       <join atomRef2Array='r1' moleculeRefs2='PARENT CHILD'>
6         <fragment homology='halogen' />
7       </join>
8     </fragment>
9     <join atomRefs2='r1 r2' moleculeRefs2='PREVIOUS NEXT' />
10    <fragment ref='oxygenOrNitrogen' />
11    <join atomRefs2='r1 r2' moleculeRefs2='PREVIOUS NEXT' />
12    <fragment ref='carbonyl' />
13    <join atomRefs2='r1 r2' moleculeRefs2='PREVIOUS NEXT' />
14    <fragment smiles='R1C1C(R2)C(R3)[C[1-4]]C1'>
15      <join atomRefs2='r1 r1' moleculeRefs2='PARENT CHILD'>
16        <fragment ref='alkoxy' />
17      </join>
18      <join atomRefs2='r3 r1' moleculeRefs2='PARENT CHILD'>
19        <fragment ref='alkoxy' />
20      </join>
21    </fragment>
22  </fragment>
23  <fragmentList>
24    <fragment id='benzene'>
25      <molecule ref='g:benzene' />
26    </fragment>
27    <fragment id='o'>
28      <molecule ref='g:o' />
29    </fragment>
30    <fragment id='n'>
31      <molecule ref='g:nsp2' />
32    </fragment>
33    <fragment id='oxygenOrNitrogen'>
34      <fragmentList role='markushMixture'>
35        <fragment ref='o' />
36        <fragment ref='n' />
37      </fragmentList>
38    </fragment>
39    <fragment id='carbonyl'>
40      <molecule ref='g:carbonyl' />
41    </fragment>
42    <fragment id='alkoxy'>
43      <fragment template='alkoxy' minC='1' maxC='4'
44        branched='true' />
45    </fragment>
46  </fragmentList>
47 </fragment>

```

Figure 3-19: EPML representation of the example Markush structure

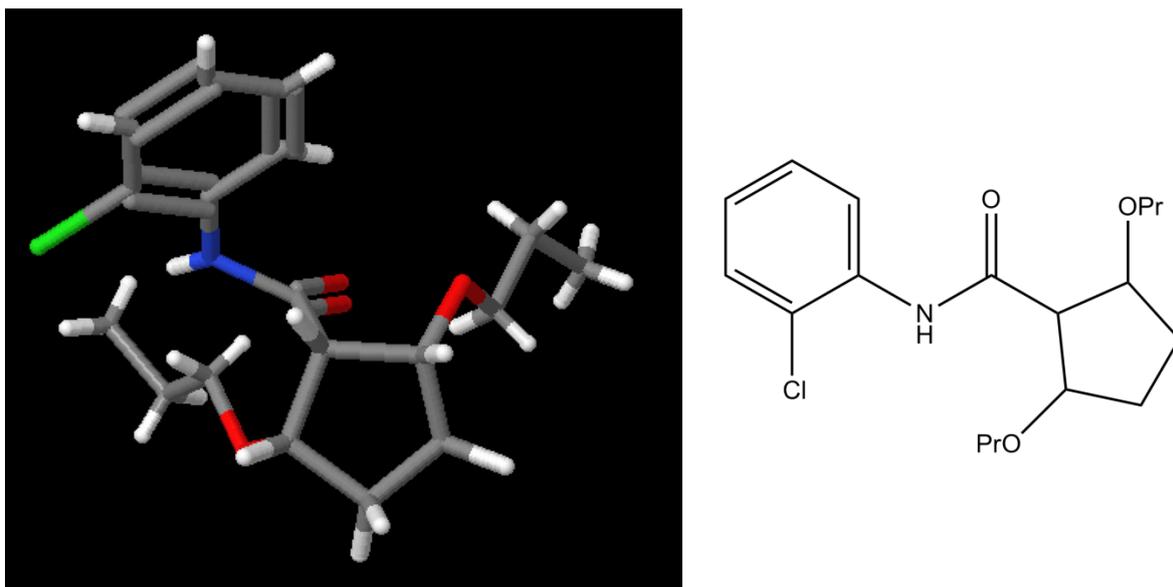


Figure 3-20: 3D (left) and 2D (right) views of a randomly-generated example compound

3.5 Substructure Searching of Markush Structures

Markush structures typically represent simultaneously a number of specific chemical structures with a shared substructure, and may therefore be considered as a superposition of the specific structures they represent. Such superpositions may be visualised as in Figure 3-21, in which the elements that are conserved and those that vary across the set of specific compounds are shown in black and red respectively.

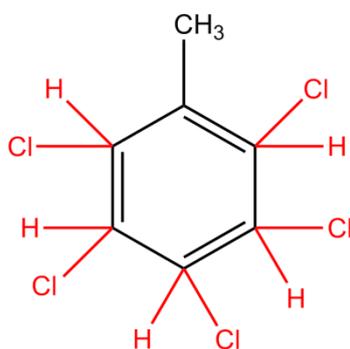


Figure 3-21: Superimposed structure representing the monochlorinated toluenes

Such Extended Connection Tables (ECTs) may be used for substructure searching, in which case the query may be asking one of two questions. If the query structure for the substructure search represents a complete chemical structure, then a hit signifies that the query structure is one of the specific compounds represented by the Markush structure defined by the ECT, while if the query structure is an incomplete chemical structure then a hit signifies that one or more of the underlying specific structures contains that substructure. Of course, such searching of ECTs must be subject to certain constraints to ensure that the correct solutions are reached – for example, no specific structure represented by the ECT above contains more than one chlorine atom and no carbon atom in the specific structures is pentavalent, in spite of such substructures being superficially present in the ECT.

The use of ECTs to represent Markush structures has been previously discussed (86) and strategies for their substructure searching developed. Previous work has described the atom-by-atom searching (87) of ECTs, as well as the use of bitscreens (88) and reduced chemical graphs (89) as methods of filtering by which to reduce the computer time required to calculate search results. In order to demonstrate the possibility of performing substructure searching of Markush structures encoded in EPML, a system was implemented that first builds ECTs from EPML documents then employs the relaxation algorithm for atom-by-atom searching as described by von Scholley (87). The details of this system are subsequently discussed.

3.5.1 Implementing Extended Connection Tables

The implementation of Extended Connection Tables used in the current work was created from scratch for the purpose. ECTs are represented by the ECT class and modelled as graphs, comprising of edges (bonds) and nodes (atoms and pseudoatoms). Nodes are represented by the abstract class Node, of which there are three implementations – AtomNode, TemplateNode and AlkylenNode.

The `AtomNode` class represents a specific atom, while the `AlkyleneNode` class represents an alkylene (e.g. $-\text{CH}_2-$, $-\text{CH}_2\text{CH}_2-$, etc.) chain and the `TemplateNode` class represents the connection table of a template used to describe homology variation. Edges are represented by the `Edge` class, which record the connectivity of the graph, and the order of the bonds which the edges represent.

3.5.1.1 Node

The abstract class `Node` exists to provide functionality that is required by all types of `Node`. Specifically, it keeps track of which edges contain the current node, which other nodes are ligands of the current node, which ECT the node belongs to, the id of the `Node`, the `markushId` of the `Node` (if any) and the maximum number of variable edges that may be simultaneously connected to the `Node`. This final property is necessary in the case of multiple position variation to enforce a limit on the number of substituents present at a single position.

3.5.1.2 AtomNode

The `AtomNode` class is intended to represent an atom of a specific type and is additionally used to represent the R-groups that indicate free valencies. In addition to the information stored by all nodes, an `AtomNode` records the element type of the node.

3.5.1.3 TemplateNode

The `TemplateNode` class represents a template as used to represent a class of substituents in homology variation and keeps track of the parameters used to specify the template *i.e.* the minimum and maximum number of carbon atoms permitted. The connection table of the template is stored as an internal ECT of the `TemplateNode`, and when a `TemplateNode` is added to an ECT

the internal nodes are added to the parent ECT to simplify the representation of the connectivity of the ECT.

3.5.1.4 AlkyleneNode

The `AlkyleneNode` class represents an alkylene chain as used in homology templates. It holds atom nodes internally to represent the carbon and hydrogen atoms that comprise the chain and records the connectivity between the atoms. It stores the ligands of the first carbon in the chain separately from those of the final carbon in order to facilitate substructure searching.

3.5.1.5 Edge

The `Edge` class represents a bond between two nodes in the ECT. It keeps track of the order of the bond, of the nodes that the edge connects and of whether or not the edge is variable, *i.e.* if it is not present in all example compounds of the Markush structure represented by the ECT.

3.5.2 Building Extended Connection Tables

The capacity to construct an ECT representing a Markush structure described by an EPML document is provided by the `Epm1Parser` class. This process is broken down into a number of steps which are subsequently described;

1. Those `fragment` elements in the source document that refer to substructures defined elsewhere in the document are modified to directly include this content. To achieve this, a copy is made of the referenced content which then is used to replace the reference.

2. Those `fragment` elements in the working document that carry `countExpression` attributes are selected by XPath. Those that specify a range, *i.e.* those that describe frequency variation, are expanded into a `markushMixture` in which the substructures corresponding to each of the permitted values in the range is represented separately.
3. Those `fragment` elements in the working document that employ inline connection tables are identified by XPath. Those that employ variation within the inline connection tables are fully enumerated and a `markushMixture` is constructed that represents each of the permitted substructures as a separate inline connection table. This `markushMixture` then replaces the original `fragment` element in the working document.
4. Those `fragment` elements in the working document that employ inline connection tables are identified by XPath, including those generated in the previous step. The inline connection tables are built into CML molecules using the `SMILESTool1` class from JUMBO, which are then appended as children to the original `fragment` elements.
5. Those `fragment` elements in the working document that carry `homology` attributes are selected by XPath. For each such fragment, the corresponding CML molecules are loaded from the homology dictionary and used to create a `markushMixture` that defines the permitted substructures, which is then appended to the original `fragment` element.
6. Those `fragment` elements in the working document that carry `template` attributes are selected by XPath. For each such `fragment`, the corresponding definitions from the template dictionary are loaded and these CML molecules are appended as children to the original `fragment` elements.
7. Those `join` elements in the working document that carry `countExpression` attributes, *i.e.* those that describe multiple substitution of a PARENT substructure, are selected by XPath. For each such `join`, the `countExpression` attribute is removed and a

markushMixture is created in which each of the permitted counts are represented by a copy of the parent `fragment` of the `join` carrying the specified number of copies of the `join`. This `markushMixture` then replaces the parent `fragment` of the `join` to produce an enumerated list of the allowed substituted fragments. For example, the fragment;

```
<fragment ref="benzene">
  <join countExpression="range(1,2)">
    <fragment ref="methyl" />
  </join>
</fragment>
```

would be converted to;

```
<fragmentList role='markushMixture'>
  <fragment ref='benzene'>
    <join>
      <fragment ref='methyl'>
    </join>
  </fragment>
  <fragment ref='benzene'>
    <join>
      <fragment ref='methyl'>
    </join>
    <join>
      <fragment ref='methyl'>
    </join>
  </fragment>
</fragmentList>
```

8. All `fragment` elements in the working document are selected using XPath. These elements are assigned unique `id` attributes, numbered from 0 to n . `Molecule` elements that are contained within `markushMixtures` are assigned a unique `markushId` attribute of the format "x_y" where x is the `id` attribute of the parent `fragment` of the `markushMixture` and y is a unique `id` number from 0 to n as before. These `markushIds` are vital to the correct searching of the ECTs as they are used to keep track of which nodes are not permitted to appear simultaneously in an example structure.
9. Those `fragment` elements with a `countExpression` attribute, *i.e.* those that indicate a repeated substructure such as $-(\text{CH}_2)_n-$ are selected by XPath and expanded by copying the

content of the fragment the number of times specified by the countExpression and replacing the original fragment with this enumeration.

10. The molecule elements in the working document are assigned unique id attributes in the same manner as was previously done for the fragment elements. A fragmentRefs2 attribute is then added to each join element that contains the unique ids of the two fragment or molecule elements that the join connects to facilitate adding the edges defined by the join elements when building the ECT.

Once these transformations have been carried out on the working document, it has been transformed into a format from which an ECT may be constructed.

1. A list is compiled of all molecule elements in the working document that descend from the primary fragment. Those that contain atomistic descriptions, *e.g.* those derived from homology variation, are directly added to this list, while those that contain references to other CML documents are dereferenced, and the atomistic descriptions are added to the list.
2. The connection tables contained in the list generated in step 1 are added in turn to the ECT. Each atom from these connection tables is represented by a Node as described previously, while each bond is represented by an Edge.
3. The edges represented by the join elements in the working document are added to the ECT. Those join elements that represent edges between two specific atom nodes result in one Edge in the ECT, while those that connect to a markushMixture or that represent position variation result in sufficient edges to connect to all members of the markushMixture or all permitted connection points respectively. Such edges are marked as *variable*.

4. Nodes to which a variable edge is connected have an additional atom node representing a hydrogen atom added as a ligand to represent the hydrogen atom that occupies the position if the variable substituent is not attached in that position. The edge to this hydrogen is similarly marked as *variable*.

Upon the completion of this process, the `Epm1Parser` has constructed an ECT that represents the superposition of the explicit structures represented by the Markush structure as described by the EPML input document. Such an ECT may then be searched to determine if it contains a specific example structure or substructure using the relaxation algorithm, as described in the next section. The API of the ECT-related classes also permits a user to construct his own ECTs programmatically, and convenience methods are also provided in the ECT class to allow a user to construct ECT representations directly from SMILES strings and from CML fragments and molecules.

3.5.3 The Relaxation Algorithm

The relaxation algorithm is a simple, iterative means by which a target structure may be determined to contain or not to contain a query structure. The basic method may be summarised as follows;

1. Assign to each atom in the query structure a unique label.
2. Assign to each atom in the target structure a set of the labels containing each of the labels from the query structure corresponding to each of the query atoms that the target atom could correspond to in a match between the two structures.
3. Iterate through the atoms of the target structure. For each label on each atom, check that the ligands of the target atom carry all of the labels carried by the ligands of the query atom that carries the label in question. If this condition does not hold, remove the label.

4. Repeat step 3 until either no labels remain on the target structure or a complete iteration through the target atoms results in no labels being removed.

If step 4 results in an unlabelled target structure, then the target does not contain the query structure. If it results in a target structure in which each label from the query structure is carried by one and only one target atom then the target structure contains the query structure. If the relaxation procedure results in any other stable state then the target structure may or may not contain the target structure.

This process is illustrated in Figure 3-22, in which at each step the labels on the atom undergoing inspection are shown in blue. Initial labelling is carried out by assigning to each target atom all labels carried by a query atom of the same element type. In the first step of relaxation, the label "4" is removed from the leftmost carbon as in the query structure the atom labelled "4" neighbours atoms labelled "5" and "6", and these labels are not carried by a ligand of the leftmost carbon atom, while the conditions to retain the labels "2" and "3" are met. The process continues until a situation is reached in which each label from the query structure occurs once and only once in the target structure, and a complete iteration through the target atoms results in no labels being removed as shown, demonstrating that a match has been found.

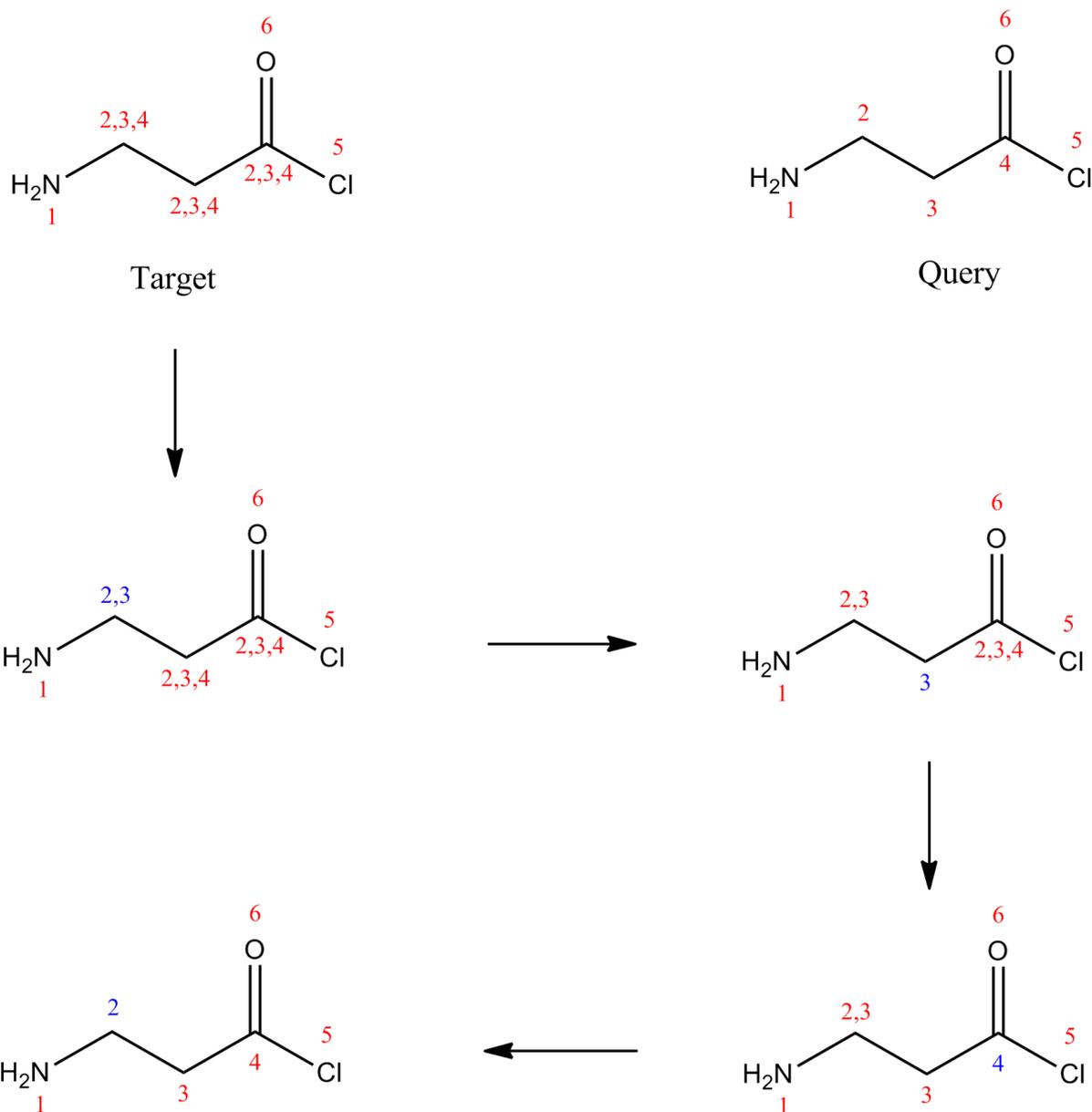


Figure 3-22: Relaxation match of 3-aminopropanoyl chloride

As mentioned previously, the algorithm outlined above is not guaranteed to produce a conclusive answer. For certain combinations of query and target structures, a stable state can be reached in which unambiguous mapping between query and target atoms has not been reached, but nor has the target structure been demonstrated not to be a match to the query. Two such examples are illustrated in Figure 3-23. In the first case, the searching of the query structure cyclopropane against the target structure cyclobutane, the initial labelling is as shown in the target structure. The

relaxation process removes no labels from the target structure, since the requirement for each label in the query structure is that it be adjacent to each of the other two and this condition is satisfied for all labels on the target structure. In the second case, the searching of dimethyl formamide against itself, the symmetry of the two *N*-methyl groups results in the relaxation algorithm terminating with both carrying the labels “4” and “5”, and so without producing an unambiguous assignment. The implementation of the algorithm used in the current work resolves such situations by selecting a node that carries more than one label, removing all but one label from the node and continuing the relaxation process. If no match is found, the process is repeated, retaining a different label until either a solution is found or it is shown that no match exists.

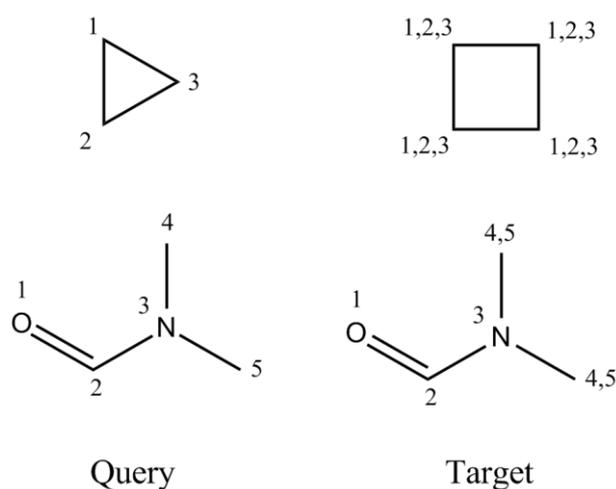


Figure 3-23: Inconclusive results of relaxation matches

The exact process used to perform relaxation matching of structural queries against ECTs in the current work is as follows;

1. Used R-groups, *i.e.* those defined in the source fragments of the EPML document that have been used to connect a further fragment to the structure are removed from the ECT. Unused R-groups are retained and later permitted to be matched to hydrogen atoms.

2. Unique labels are arbitrarily assigned to the atom nodes that make up the query ECT. No other types of nodes are permitted to form a part of the query ECT.
3. Each node in the target ECT is labelled against the query ECT. Atom nodes are given the label of a query node if they are of the same element type or the query node is hydrogen and the target node is an R-group, and if the target node has as at least as many ligands of each element type as the query node in question. Alkylene nodes that are ligands of the target atom node are considered as carbon atoms for this purpose. The length of the carbon chain embedded within an AlkyleneNode is set to be equal to the number of carbon atoms in the query structure and each internal carbon atom is labelled with all of the corresponding labels from the query structure. Each embedded carbon atom is given two ligand hydrogen atoms which are given all the labels carried by the ligand hydrogens of the query structure's carbons.
4. An initial label reduction identifies those nodes in the query structure that carry the only instance of a certain label. Where found, those nodes are stripped of all other labels, since a match may only be found in circumstances where that node carries that label. The process repeats until no such nodes are found.
5. The nodes of the target ECT are relaxed. In each iteration, a given label is removed from an atom node in the target structure that carries it unless the node's ligands carry all of the labels adjacent to the given label in the query structure and the orders of the edges that connect the candidate equivalent nodes in the target structure match those in the query structure. The procedure for the removal of labels from the embedded nodes in an alkylene node follows a similar method. The internal carbon atom nodes are held in a list in which the first is considered to be the "leftmost" and the last the "rightmost", and those nodes adjacent in the list are considered to be connected. During relaxation, for each internal carbon atom node sets of adjacent labels are computed in which one label is selected from

each of the nodes connected to the current node, the leftmost ligands if the current node is the first in the internal list and the rightmost ligands for all internal nodes. If none of the label sets so generated contains the set of adjacent labels from the query structure then the label in question is removed from the embedded target atom node. Whenever the rightmost node becomes unlabelled it is removed from the internal list, allowing the alkylene node's internal carbon chain to shrink until it is of the correct size to match the query structure or until it reaches zero length and carries no labels, indicating that the alkylene node is not involved in any potential match to the query structure. Step 5 repeats until a stable state is reached.

6. The target ECT is checked to determine whether a premature stable state has been reached. If so, a multiply labelled node is selected and n copies of the ECT, where n is the number of labels on the selected node, are created in which the selected node carries only a single label. These ECTs are returned to step 5 for further processing. If the ECT carries each query label once and only once, the candidate solution is checked to ensure its validity. It is checked that the solution does not use nodes derived from more than one member of each markushMixture using the nodes' markushIds, each TemplateNode contained in the ECT checks that its carbon count is permitted and each node is checked to ensure that its variable edge limit is not exceeded. If these checks are passed, a solution to the search has been found and is returned; if not, the ECT is discarded and the next ECT is considered.

3.5.4 Examples

To demonstrate the building and searching of ECTs, a number of examples are subsequently discussed. A simple Markush structure and a corresponding ECT are shown in Figure 3-24. In the ECT, the halogen atoms (shown in blue) bear markushIds that indicate that not more than one of them may be used simultaneously when searching, and the variable edges in the ECT – those that connect

to the variably positioned methyl group and to the hydrogen atoms added to attachment points of said methyl group – are shown in red. The alkylene node representing the alkyl chain R is denoted in the ECT by the symbol “Q” and other nodes are atom nodes of the element type shown.

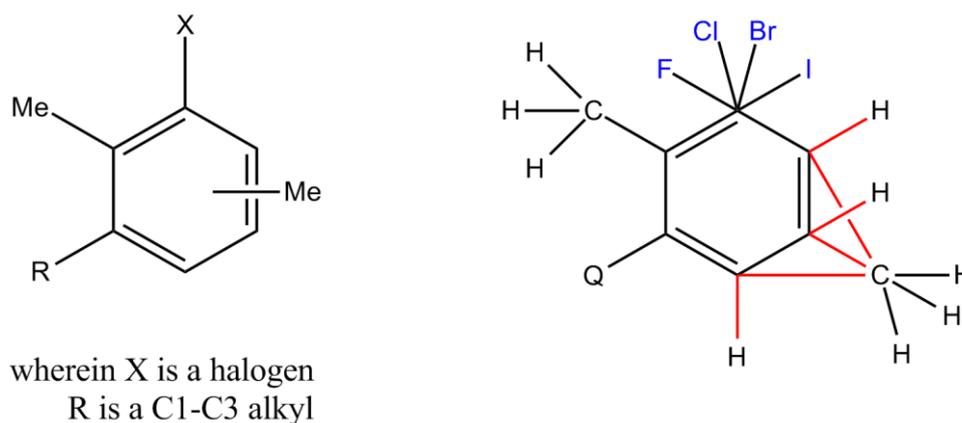
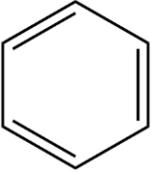
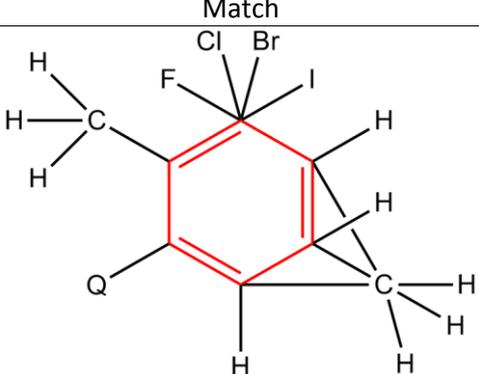
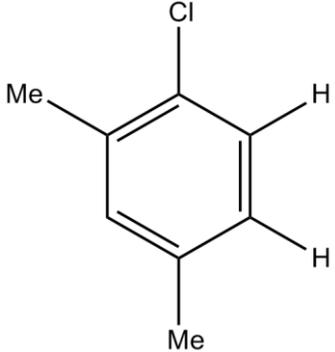
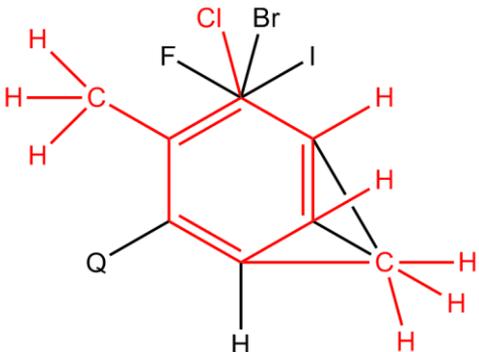
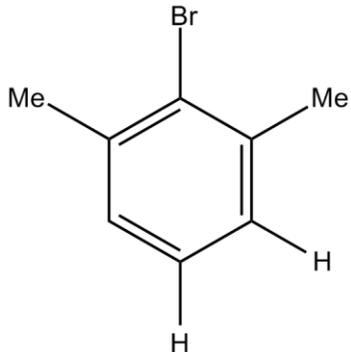
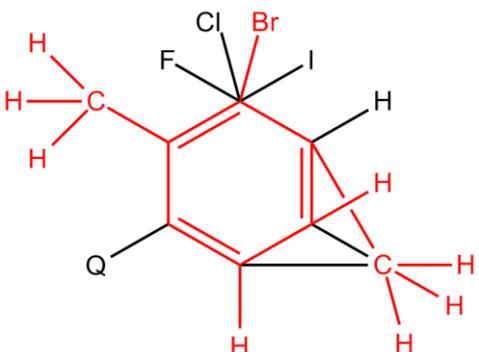
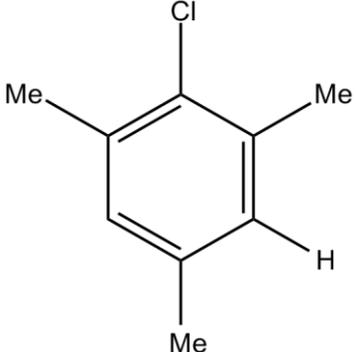


Figure 3-24: Example Markush structure (left) and corresponding ECT (right)

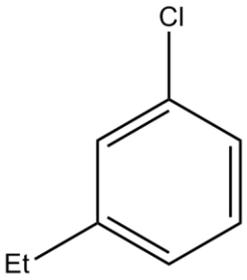
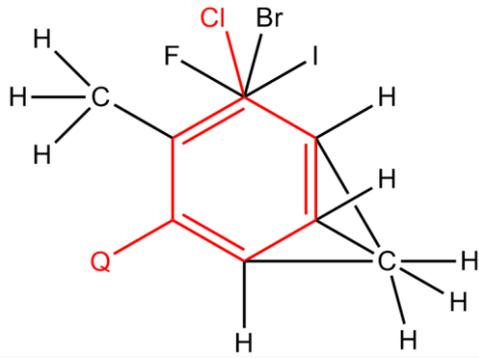
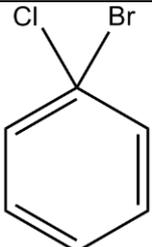
Examples of searching the ECT shown from Figure 3-24 are subsequently discussed. In each case, the query structure should not be construed to indicate the presence of any implicit hydrogen atoms beyond those that make up the methyl and ethyl substituents. When matches are found for the given query, the matching substructure of the ECT is shown in red. When discussing these examples, the atoms of the benzene ring are numbered clockwise from the top, starting at 1.

Example no.	Query	Match
1		
2		
3		

In the first example the scaffold of the Markush structure, the benzene ring, is easily identified as matching the query structure. In the second example, the methyl group in the 4-position of the query structure is matched to the methyl group with the unspecified locant, while the two hydrogen atoms in the query structure are matched to those hydrogen atoms automatically added to the ECT at positions permitted to be connection points for such mobile substituents. Similarly, this mobile methyl group may be found in the 2-position, as in the third example.

Example no.	Query	Match
4		No match
5		No match

The mobile methyl group may not, however, be found in two positions simultaneously and the search algorithm does not allow it to be simultaneously labelled for the two search methyl groups in a result; hence, the query in the fourth example does not produce a hit. Likewise, in the fifth example, while the carbon from the mobile methyl group may appear in both the 3- and 4- positions, it may not appear in both simultaneously. In this case a result that matches the query structure violates the variable edge limit of the mobile carbon, and so no hits are found.

Example no.	Query	Match
6		
7		No match

In the sixth example, the ethyl group is correctly matched to the alkylene node while in the seventh example only one of the halogen atoms is permitted to be used at a time in a search result, as enforced by the markushIds carried by the halogen atom nodes and so the query produces no hits.

3.6 Conclusions

The work presented in this chapter has demonstrated an outline implementation of a CML-based system for the representation and manipulation of Markush structures which supports the major features of Markush structures as they are used in the patent literature. Polymer Markup Language, an existing CML vocabulary, has been extended to permit the more convenient description of Markush structures and systems have been developed to enable the creation of example molecules and substructure searching of the Markush structures.

While the current work has shown how it is possible to produce machine-understandable Markush structures that are compatible with the semantic web of chemistry, it is in need of further

development before it can be said to be a competitor to the currently available commercial systems. In particular, the implementation of substructure searching provided by the current system does not have an understanding of aromaticity or of stereochemistry – features which would likely be required in any production system.

Since the commencement of the current work, ChemAxon have begun to add functionality to their software, Marvin, to permit the drawing, representation and manipulation of Markush structures (90) and have further demonstrated the potential for the automatic conversion from the Markush DARC format to their own. While the Marvin format is loosely based upon CML, it must be considered essentially to be another proprietary format that is not suitable for the semantic web of chemistry. It is encouraging, however, to see that the automatic conversion of Markush formats has been shown to be possible as it allows for a transition from the current situation without the need to abandon the data that has collected up to the current point.

4. Automatic Acquisition of Hyponymic Relations from the Chemical Literature

When considering a problem, the human thought process makes much use of background knowledge, whether in the chemical sciences or in everyday life. Chemical Markup Language provides a means to describe defined chemical concepts such as molecules, reactions and spectra but does not give the freedom to define the relationships between novel concepts. In the Semantic Web, this capacity is provided by ontologies – and much effort has been devoted to their construction. The most basic elements of these formal representations of knowledge are a set of hierarchical hyponymic relations – descriptions of which concepts form supersets or subsets of which other concepts which may be considered at the most basic level as dictionaries of terms. This chapter considers how these dictionaries may be automatically derived, using the published literature as a source of knowledge, and discusses some of the applications to which the derived knowledge may be put.

4.1 Hyponymic Relations

Hyponymic (“is-a”) relations exist between two terms where one the hyponym, is a subset of the other, the hypernym. For example, “vehicle” is a hypernym of “car”, and “Ford Fiesta” is a hyponym of “car”. Knowledge of such relations forms a key part of the way that we reason and form deductions. Consider, for example, the following statements;

- i. Reactions between carboxylic acids and alcohols that form esters are esterifications.
- ii. Ethanol is an alcohol.
- iii. Acetic acid is a carboxylic acid.

- iv. Ethyl acetate is an ester.

It therefore follows that the reaction between acetic acid and ethanol that forms ethyl acetate is an esterification. In order to facilitate the automation of such, and other, reasoning – the ultimate goal of the semantic web – it is necessary first to encode the starting axioms in a formal representation such as an ontology. The creation of a formal knowledge representation is typically a slow process as manual curators determine and verify the information to be curated, and such work is confined to fields that the curators consider to be within the scope of their work. This work may be both quickened and broadened by the automatic acquisition of the hyponymic relations.

4.2 Hearst Patterns

Hearst first proposed the use of lexico-syntactic patterns for the automatic acquisition of hyponymic relations (91), thereafter known as *Hearst Patterns*. She described six patterns that could be employed;

Format	Example	Pattern Name
<i>HYPER</i> such as <i>HYPO</i>	Apolar solvents such as THF and hexane	SUCH_AS
such <i>HYPER</i> as <i>HYPO</i>	Such bases as NaOEt or LDA	SUCH_FOO_AS
<i>HYPO</i> or other <i>HYPER</i>	MeCl, EtBr or other organohalides	OR_OTHER
<i>HYPO</i> and other <i>HYPER</i>	Benzene, ethylene oxide and other carcinogens	AND_OTHER
<i>HYPER</i> including <i>HYPO</i>	Methyl ketones including acetone	INCLUDING
<i>HYPER</i> especially <i>HYPO</i>	Grignard reagents, especially methyl magnesium chloride	ESPECIALLY

Table 4-1: Hearst Patterns and their usage in chemical texts

wherein *HYPER* and *HYPO* represent noun phrases that denote the hypernym and hyponym(s) respectively. Each of these patterns has been assigned a name for ease of reference.

Taking the example for the SUCH_AS pattern, it can be seen that the text communicates the information that THF and hexane are examples of apolar solvents. This information is readily

available to any fluent speaker of the English language, regardless of whether or not they are aware of what “THF”, “hexane” or an “apolar solvent” are. The application of these six patterns to a large corpus of text therefore provides a powerful method for the identification of hyponymic relations between lexical terms, and the approach has been broadly applied, including in the biomedical sciences (92) – though not previously to the chemical sciences.

Of course, chemical documents contain Hearst Patterns that are not relevant to the chemical field. In order to limit the domain of hyponymic relations identified by any automated system, it is necessary to apply some form of filtering. The capacity to separate “chemical” from “non-chemical” words is provided by the OSCAR3 toolkit, as previously discussed. Indeed, a basic approach to the problem is implemented within OSCAR3 itself.

4.2.1 OSCAR3 Implementation

The OSCAR3 application of Hearst Patterns employs token-based regular-expression style matching in the style as employed in the PatternRecogniser (see section 2.2.6.3), and uses as a core feature the named-entity recognition and the experimental subclass classification provided by OSCAR3. The SUCH_AS pattern described above, for example, is specified by OSCAR3 as;

```
$CM:CLASS<hyper> $MAYBECOMMA $SUCHAS $CMEXACTHYPO
```

\$CM:CLASS refers to a named entity of type CM (Chemical – see section 2.2.6.3) and of subclass CLASS, while the meanings of \$MAYBECOMMA, \$SUCHAS and \$CMEXACTHYPO are defined separately as follows;

```
$MAYBECOMMA = $( , $) $?
```

In this definition, the \$ symbol before the brackets and the question mark symbol indicate that they are to be interpreted as in standard regular expressions *i.e.* the brackets define a character group

while the question mark states that the preceding group is optional, while the comma matches a literal comma. Thus, the \$MAYBECOMMA expression matches either one comma or nothing at all.

```
$CMEXACTHYPO = $( $( $CM:EXACT<hypo> $)
                  $|
                  $( $( $CM:EXACT<hypo> , $) $*
                    $CM:EXACT<hypo> $MAYBECOMMA $ANDOR
                    $CM:EXACT<hypo> $)
                  $)
```

```
$ANDOR = $( and $| or $)
```

The whole expression for \$CMEXACTHYPO thus matching either a single named entity of type CM and subtype EXACT, or a comma-separated list of arbitrary length terminated by “and” or “or” and a final CM:EXACT, *e.g.* “methane, ethane, propane, butane and pentane”, or simply “methane”.

```
$SUCHAS = $( such as $| including $| excluding $|
             particularly $| especially $| mainly $| primarily
             $| chiefly $| specifically $)
```

Here we see a number of Hearst Patterns, including the SUCH_AS pattern described above, being implemented at once – thus widening the scope for hyponymic relation acquisition. A number of other Hearst Patterns are similarly implemented by OSCAR3, and they are not confined to the field of hyponymic (“is-a”) relations, but also “has-a”, *e.g.* “the ester carbonyl” – indicating that a carbonyl is a part of an ester.

Because the OSCAR3 implementation depends on a token-level match it lacks some of the flexibility that is afforded by the standard natural language approach of chunking the tokens into phrases and identifying hypernyms and hyponyms as noun phrases. The hypernym is required to be of type CM or of a specific subtype of CM, causing useful relations to be ignored – the class CM is intended to refer to words that have structural or substructural meaning. As a result, hypernyms based on usage, function or properties, *etc.* are ignored, for example the relation “bases such as LDA and *n*-butyl lithium”. This capability could be added by selecting a number of hypernyms that, while not

classified as CM by OSCAR3, are of sufficient chemical importance to be included, but this approach would not be truly unsupervised, and would therefore miss subtleties in the text *e.g.* “strong bases such as LDA and n-butyl lithium” or “5-HT_{1A} antagonists such as Lecoztan or Spiperone”, in which the crucial function of the drugs is not they are antagonists, but that they are antagonists of the 5-HT_{1A} receptor.

4.3 Acquiring Hyponymic Relations

In order to address the issues described in the previous section, a new system was developed to apply OSCAR3’s name recognition capabilities to the problem of the detection of Hearst Patterns in chemical texts. This system aims to carry out unsupervised detection of molecular classifications, such as those exemplified in Table 4-1, and is based around ChemicalTagger (described in section 2.2.7) as outlined below;

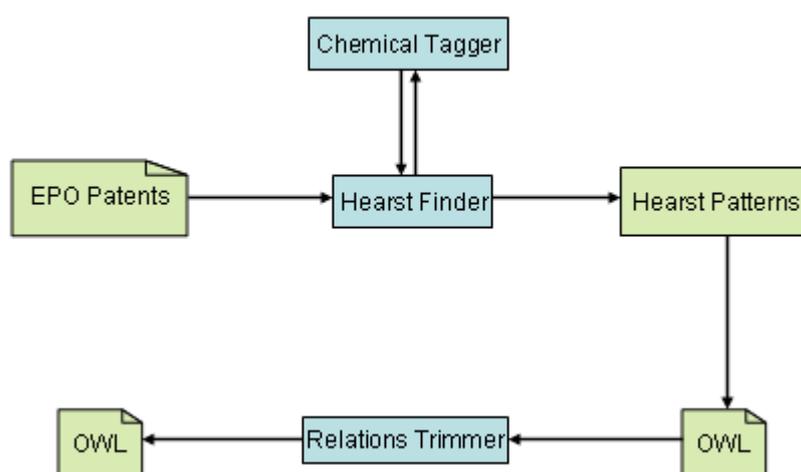


Figure 4-1: Acquisition and Storage of Hearst Patterns

In this system, the text from the EPO patents is passed into the `HearstFinder` one paragraph at a time. The `HearstFinder` then uses `ChemicalTagger` to analyse the grammar of the input text – returning a tagged and chunked document. From these documents it is possible to identify sections of the input text that correspond to a given Hearst Pattern, *i.e.* the noun phrases that denote hypernym and hyponym as well as the invariant text characteristic of the Hearst Pattern *e.g.* “such as”. By checking that the hyponym phrase consists of a sequence of entities of type CM, it is possible to narrow the set of identified hyponymic relations to those that define relations between chemical structures. The set of hyponymic relations derived from this process are then formally encoded into the Web Ontology Language (OWL) (93). Using appropriate heuristics, these relations are then trimmed with the intention to remove unreliable or inaccurate assertions. The full process is subsequently discussed in detail.

4.3.1 HearstFinder

The application of Hearst Patterns to and identification of hyponymic relations within an input text is handled by the `HearstFinder` class. The operation of this class is subsequently discussed.

4.3.1.1 Specification of Hearst Patterns

Hearst Patterns to be identified are defined to the `HearstFinder` as space-separated string representations of literal and meta-tokens, for example, the `SUCH_AS` pattern is defined as “\$HYPER such as \$HYPO”. The `HearstFinder` treats “\$HYPER” and “\$HYPO” as meta-tokens that denote the location of the noun phrases that define the hypernym and hyponym respectively, while other tokens in the specification are treated as literal tokens that define the structure of the Hearst Pattern and must be present in the source text in the same positions relative to the noun phrases as

they are in the specification relative to the meta-tokens in order for the source text to be considered to match the Hearst Pattern. The matching of phrases with meta-tokens is illustrated in Figure 4-2.

This approach offers a greater flexibility than that of the OSCAR3 system since it allows for a user to easily apply the existing code to novel Hearst Patterns and is not limited to the discovery of pre-defined or recognisably chemical hypernyms and hyponyms. Indeed, the sections of input text that are identified by the HearstFinder are not in the first instance restricted to the chemical domain in any way. The software is therefore potentially reusable in a general context.

4.3.1.2 Matching Hearst Patterns

Initially, the input text is passed to ChemicalTagger for grammatical analysis. The target Hearst Pattern is specified as described above and passed to the HearstFinder as an argument. Instances of this pattern are identified by locating instances of the first literal token from the target pattern within the text and simultaneously advancing through the tokens of the target pattern and source text. When a literal token is found in the target pattern, this token must be the next token of the source text; when a metatoken is found in the target pattern the containing phrase is noted and the pointer that tracks which position in the text is being examined is advanced to the end of this phrase. If this process can be completed successfully then metatokens in the target pattern that occur prior to the first literal token are matched by looking backwards in the source text from the position of the match to the first literal token. For example, the input text “Apolar solvents such as THF and hexane may be employed” produces the grammar tree shown in Figure 4-2, wherein NP indicates a noun phrase, VP indicates a verb phrase and PP indicates a prepositional phrase.

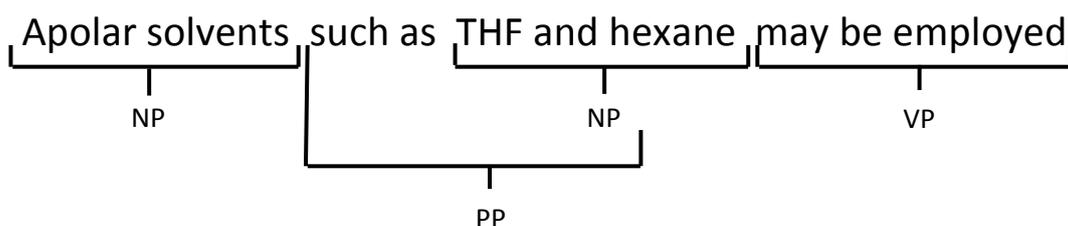


Figure 4-2: Grammatical structure of a Hearst Pattern

When the SUCH_AS pattern (\$HYPER such as \$HYPO) is applied to this result, the hypernym and hyponyms are identified as the immediate parent phrases of the tokens before and after the literal pattern text ("such as"), *i.e.* "Apolar solvents" and "THF and hexane" respectively. At this stage, in order to eliminate non-chemical hyponymic relations *e.g.* "large cities such as Tokyo and London", the content of the hyponym is checked using OSCAR3; each of the tokens making up the text of the hyponym must either have been tagged as CM by OSCAR3, or must belong to a predefined list of allowable tokens including articles ("a", "an" and "the"), appropriate punctuation (comma and semicolon) and conjunctions ("or" and "and").

4.3.2 Recording Hyponymic Relations

A hyponymic relation defines one concept to be a subclass of another. Since such relationships perform a key role in ontologies – formal representations of knowledge – there are existing tools that provide a means for their storage and manipulation. In the current work, hyponymic relations are converted to, and stored in, the Web Ontology Language (OWL) (93). The required functionality for reading, writing and creating OWL is provided by the open-source OWL API library (94), version 3.0.0.

Each extracted hyponymic relation is represented by a SubClassOf axiom in OWL, which denotes that one class – the subclass – is a subset of another – the superclass. Both hypernyms and hyponyms are

represented by OWL classes. Hypernyms are represented by unique classes if the text string composing the hypernym phrase is novel once it has been lowercased and stripped of leading determiners and the terminal character 's', as a naïve approach to the problem of pluralisation. Thus, the hypernyms "Esters", "some ester" and "an ester" would be considered equivalent and represented by the same OWL class.

Individual chemical names have already been identified within the hyponym phrase by `ChemicalTagger`, and the `MOLECULE` elements therein are taken to be individual hyponyms. It is not desirable to create multiple OWL subclasses for a single chemical substance, so non-novel hyponyms are identified both by string equivalence and by resolution of chemical names to InChIs using the `NameResolver` class from OSCAR3. The resulting connection table is converted to InChI using the `InChIGeneratorTool` class from JUMBO. If the resulting InChI has been previously seen then the class representing this structure is used as the subclass for the new axiom. Using this method, the hyponyms "chloroform" and "trichloromethane" are represented by the same OWL class. Since the current intent is to derive classifications of chemical compounds from the patent texts, relations for which it is not possible to resolve the hyponym to a connection table and to an InChI are discarded. This may occur, for example, in situations where the hyponym is systematic nomenclature that is not supported by OPSIN, a trivial term that does not occur in OSCAR's dictionary or a term denoting a chemical fragment, such as "ethyl".

The OWL file created by this method contains more information than purely which terms are hyponyms of which other terms. The OWL classes corresponding to the hyponyms are annotated to include all the identified synonyms for a structure and the InChI for that structure, as well as the text of the paragraph(s) from which the hyponymic relation was inferred. Access to the source text is vital for the identification of sources of error in this procedure, as will be seen later. In addition, the `SubClassOf` axioms are annotated with the number of patents from which the relation has been inferred. Since relations that are asserted by a large number of sources may be considered to be

more reliable or more important than those asserted by fewer, this allows for the elimination of unreliable and unimportant information at a later stage.

4.3.3 Content of the Derived Relations & Sources of Error

The full texts of the patents from the corpus of 667 unique, full-text patent documents (collected as described in section 5.1.3) were passed through the system. One of these patents, EP1651230, was found to contain text that caused ChemicalTagger to freeze and so was omitted from the procedure. The system identified 5624 Hearst Patterns across the remaining 666 patents, with each patent containing between 0 and 96 individual Hearst Patterns. The distributions of Hearst Patterns across the corpus and the contributing Hearst Patterns are summarised in Figure 4-3 and Figure 4-4 respectively.

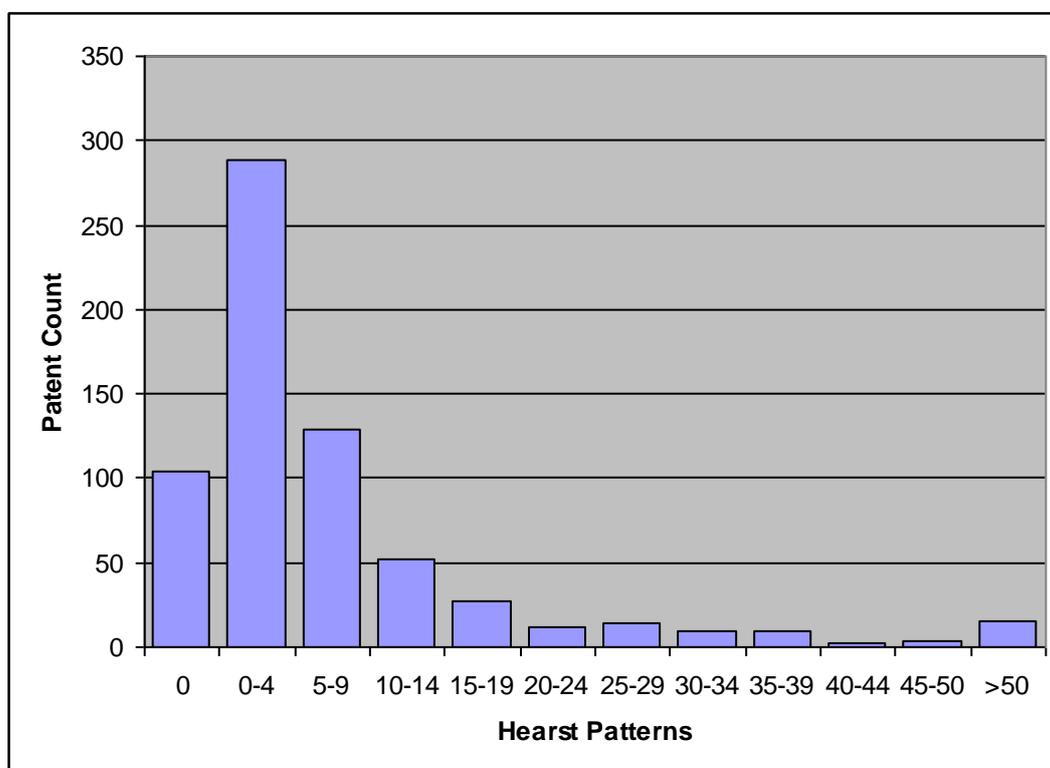


Figure 4-3: Distribution of Hearst Patterns across the patent corpus

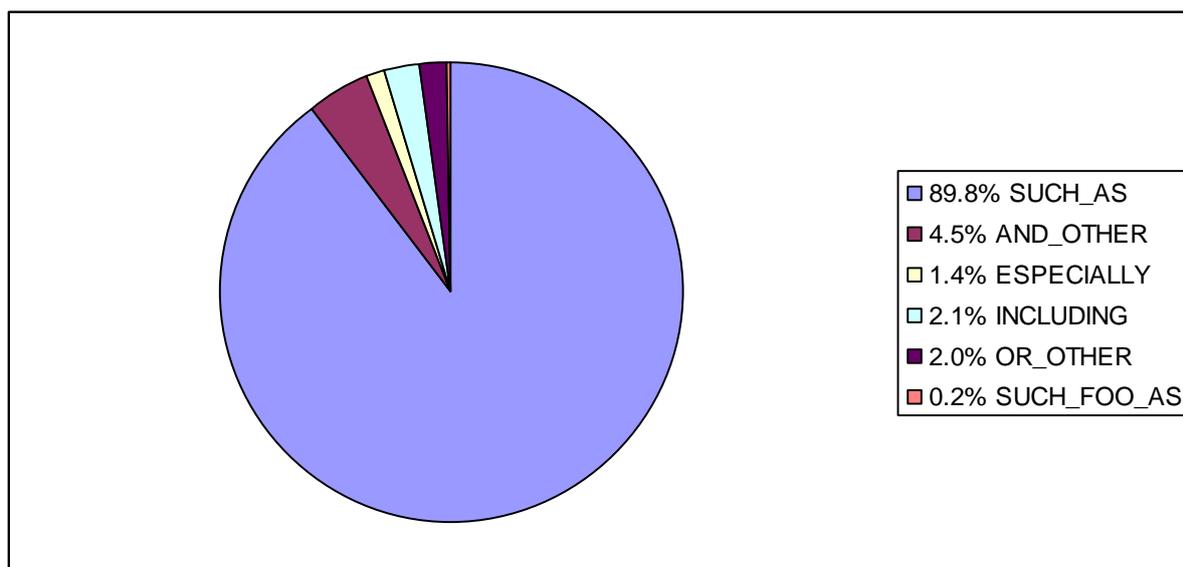


Figure 4-4: Individual Hearst Pattern frequency across the patent corpus

It is particularly noteworthy in Figure 4-4 that the SUCH_AS pattern is dominant in terms of usage, comprising around 90% of the Hearst Patterns identified within the source texts.

The OWL file derived from these Hearst Patterns defines 1001 superclasses (*i.e.* hypernyms), 984 subclasses (*i.e.* hyponyms) and a further 13 classes denoting terms that appeared as both hypernyms and hyponyms such as “pyridine” – being both a specific compound, C_5H_5N , and the class of compounds that contain a pyridine ring. Each superclass (including those that were also subclasses) had between 1 and 35 subclasses, collectively defining a total of 2738 unique hyponymic relations.

The derived hyponymic relations contain a number of sets of superclasses in which one is a superset of the others *e.g.* “base” and “strong base” or “solvent” and “chlorinated solvent”. In such cases, the superclasses are formally related only in that they may share some or all subclasses. In the chemical field it is dangerous to make the assumption that a class that fits the pattern *adjective-noun* is a subset of the corresponding class *noun*, since while a chlorinated solvent is a solvent, a chlorinated hydrocarbon is not a hydrocarbon. Consequently, the system implemented here makes no such assumptions in order to avoid introducing such errors.

Of course, automated systems make mistakes and so the hyponymic relations derived from the source texts are not perfectly reliable. It was not feasible to manually examine and validate a knowledge base of this size and so no metrics are available for the raw performance of the system on the full set of input texts, though a validation was carried out on a subset of the input text and is discussed in section 4.3.5.

The full set of derived hyponymic relations includes molecular classifications that may be considered common and generic (*e.g.* “polar aprotic solvent” and “strong base”), those that define common structural classes (*e.g.* “alcohol” and “amino acid”), those that define specific classes likely to be relevant to only a small subset of the patent documents (*e.g.* “antibiotic” and “antipsychotic drug”) and those that are entirely meaningless and have been included as a result of a mistaken parse of the input text such as those shown in Table 4-2.

Hypernym	Source text
abbreviated word	In Table 2, DHP-Cz represents 3,6-dihydroxyphenyl-9-decyl-carbazole, and other abbreviated words are the same as described in Table 1.
maybe used	A wide range of reducing agents maybe used, such as sodium borohydride, formaldehyde, formic acid, sodium formate, hydrazine hydrochloride, hydroxylamine, and hypophosphorous acid.

Table 4-2: False hypernyms and their source texts

Further, and potentially more serious, false assertions are contained where the input text lists examples of two or more molecular classes at once. For example, for the input text;

“Compounds of formula I wherein R8 is NRcRd and R9 is hydrogen may be prepared by treatment of the appropriate precursor containing the C31-C32 unsaturation with HNRcRd or HCl; HNRcRd in an appropriate protic or aprotic solvents such as methanol, ethanol, benzene, toluene, dimethylformamide, dioxane, water and the like.”

ChemicalTagger identifies the noun phrase preceding “such as” as “aprotic solvents”, resulting in the misclassification of methanol, ethanol and water as aprotic solvents. Linguistic constructions of this form are problematic to the system as it is only possible to identify which of the molecular classes the hyponyms belong to by the application of the domain-specific knowledge that we seek to identify from the source texts.

4.3.4 Trimming the Relations

In order to produce a knowledge base of manageable size and of higher quality, it was necessary to remove a number of classes and associated axioms. The full OWL file derived from the previous process was therefore trimmed based on a requirement that all axioms should have been derived from a minimum of three separate patent documents. Following the removal of axioms in this way, orphaned classes, *i.e.* those that had no remaining superclasses or subclasses, were also deleted. In this way, it was hoped that many of the mistaken classifications would be eliminated since it would be less common for a mistake to be repeated than for a correct hyponymic relation to be specified across multiple documents. It was further hypothesised that this removal of invalid axioms could be improved by increasing the number of input documents and the minimum source threshold, though this was not tested.

Following the trimming of the derived relations in this way, the resultant OWL file defined 133 superclasses and 330 subclasses. Each superclass had between 1 and 24 subclasses, collectively defining a total 516 hyponymic relations.

These 516 hyponymic relations were subjected to manual inspection and verification. Since the purpose of this exercise was to assess the validity of the hyponymic relations extracted from the source patents, hypernyms were judged to be acceptable if the extracted term was a grammatically and semantically valid description of a class of chemical structures while hyponymic relations were

judged to be acceptable if the molecule generated from the hyponym term could be correctly described as a member of the class in question. The validity of hyponymic relations was only assessed where the hypernym concerned had already been judged to be acceptable. The results of this process are summarised in Table 4-3.

Task	Acceptable	Not acceptable	Total	% acceptable
Hypernym verification	113	20	133	85.0%
Hyponymic relation verification	459	28	487	94.3%

Table 4-3: Verification of trimmed hyponymic relations

Of the retained hypernyms, 85% were judged to be acceptable according to the preceding criteria. Of the 20 hypernyms judged not to have been acceptable, 5 were found to have been included due to the incorrect interpretation of the term “methyl” as denoting the chemical structure of methane, allowing such hypernyms as “radical” and “group” to enter the molecular structure classification. One unacceptable hypernym, “dien” was included as a result of typos where “diene” was intended. One hypernym, “feedstock”, was judged to be unacceptable as the term is a description of a bulk material as opposed to of specific chemical compounds. The remaining 13 unacceptable hypernyms were generated as a result of incorrect grammatical parsing by ChemicalTagger. In 10 of these cases, ChemicalTagger included too much text in the hypernym while in 3 cases too little text was included. Examples of these cases are illustrated in Table 4-4.

Hypernym	Source text
Can be formed from a variety of phospholipids	Liposomes can be formed from a variety of phospholipids such as cholesterol, stearylamine or phosphatidylcholines.
Are typically diluted with an inert carrier	Such compositions are typically diluted with an inert carrier, such as water, before application.
Agent	...for example, sweetening agents such as fructose, aspartame or saccharin...
Inhibitor	...adenosine diphosphate (ADP) inhibitors such as clopidogrel...

Table 4-4: Unacceptable hypernyms and sample source texts

When examining the hyponyms of those molecular classes considered acceptable, a high proportion – 94% – were found to be valid examples of the hypernym concerned. The interpretation of six of the hypernyms – “non-toxic pharmaceutically acceptable inert carrier”, “suitable solvent”, “suitable base”, “pharmaceutically acceptable solvent”, “appropriate solvent” and “low molecular weight aliphatic alcohol” – was found to involve a degree of subjectivity. In these cases the acceptability of the hyponyms was adjudicated without reference to the subjective qualification – thus, hyponyms of “suitable solvent” were considered acceptable if they were solvents, *etc.* This high success rate suggests that the technique offers a very powerful means by which to identify and formalise chemical classifications.

4.3.5 HearstFinder Validation

In order to quantify the performance of the HearstFinder, an annotation task was undertaken to permit the comparison of different humans’ opinions of what constituted a chemical Hearst Pattern with each other and with the performance of the machine. Annotation guidelines were written, and are attached in Appendix A. The goal of the task was to derive performance metrics within a defined scope. Since, on average, around 8.5 Hearst Patterns were curated from each patent, it was not feasible to verify against manually annotated sections of patent text selected randomly and without

limits. Instead, it was decided to focus the task on the SUCH_AS pattern, which provided 90% of the curated hyponymic relations, and to select the corpus from among those paragraphs that contained the string “such as”. Since the HearstFinder implementation requires the inclusion of the invariant text of a Hearst Pattern in the input text in order to return any results, this filtering of the corpus paragraphs served only to remove a large proportion of paragraphs that would be of no interest to a manual annotator and could not affect the performance metrics. The limitation of the task to the SUCH_AS pattern allowed the filtering to produce a corpus in which the relevant content was highly enriched, and the exclusion of the other patterns – which collectively contributed only 10% of the total curated relations – was considered an acceptable trade-off.

The annotation guidelines are intended to cause the annotators to behave in the way that HearstFinder is intended to operate. The annotator is free to identify the hypernym according to his or her judgement of what is correct, while hyponyms are required to be terms that have structural meaning – though not artificially limited solely to those classed as CM by the OSCAR3 annotation guidelines. The annotated patterns thereby produced consist of those that fit the form that the HearstFinder is intended to identify, while the manually annotated hypernyms and hyponyms allow for the validation of those that the system automatically annotates against what a human considers to be the correct term.

The corpus for this exercise was assembled by randomly selecting 300 paragraphs of text that contained the phrase “such as” from the set of unique, full-text patent documents assembled as described in section 5.1.3. A further, non-overlapping, set of 30 paragraphs selected by the same method were used in advance of the full annotation task to ensure that the annotators understood and could implement the annotation guidelines. Three annotators were used for the task; the present author, an academic with many years’ experience of the chemical domain and a summer

student who had completed two years' undergraduate study of chemistry¹. Each annotator's results were compared to those of the other annotators and to those of the machine.

Human annotation of the corpus was carried out using a customised version of the OSCAR3 ScrapBook functionality. The OSCAR3 ScrapBook allows a user to manually annotate the OSCAR3 named entities within a sample text using a web browser, by selecting the text to be annotated and clicking the button associated with the desired named entity class. By replacing the OSCAR3 named entities with the named entities "pattern" "hyper" and "hypo", a tool for annotating the Hearst Patterns within the corpus was created. This tool is shown in Figure 4-5, while the XML annotations produced thereby is shown in Figure 4-6.

¹ In the presentation of the results, "annotator A" was the current author, "annotator B" was the summer student Shaoming Chen and "annotator C" was the academic Peter Murray-Rust. All three sets of annotations are available for inspection on the attached disk.

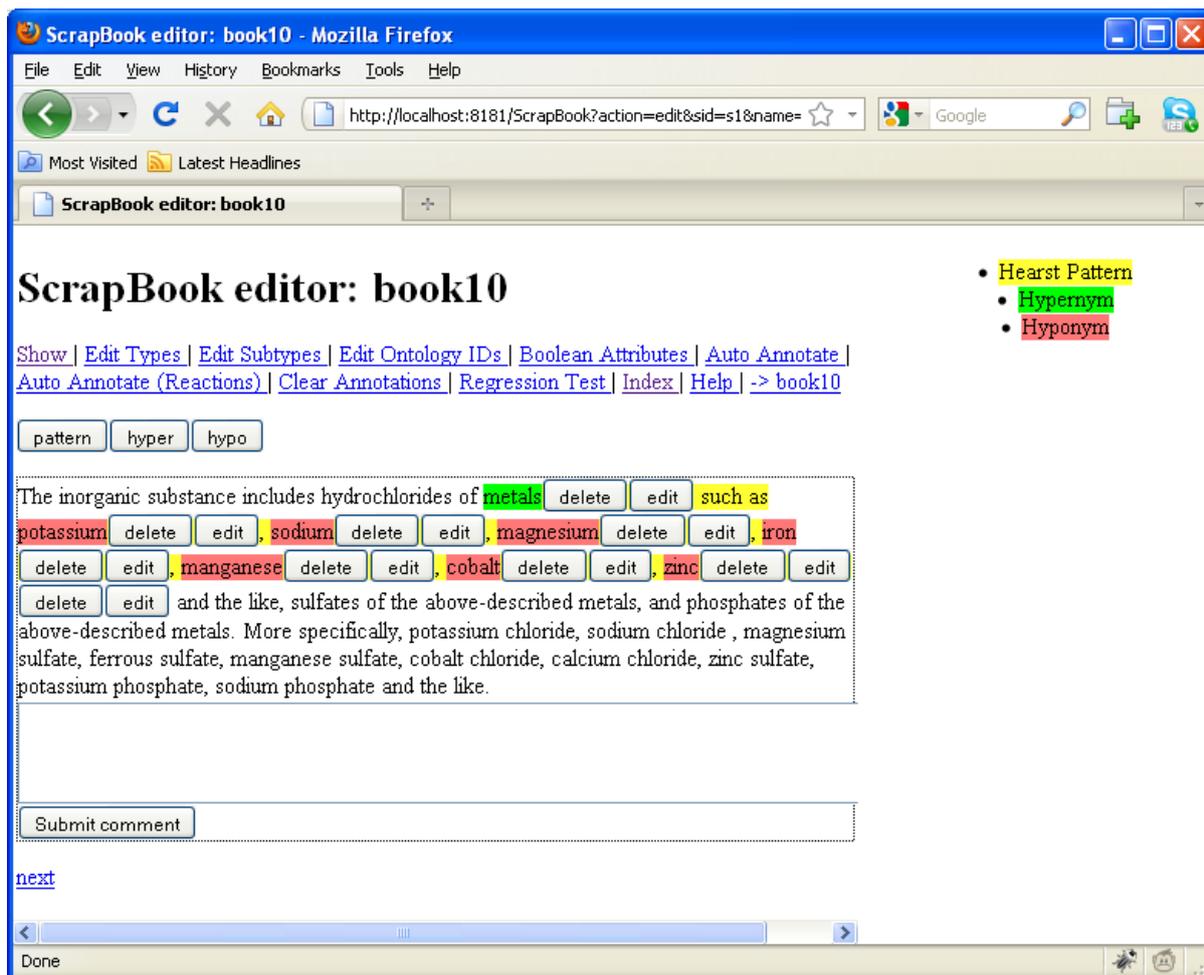


Figure 4-5: The customised OSCAR3 ScrapBook

```

<P>
  <snippet id="s1" fileno="p0">
    The inorganic substance includes hydrochlorides of
    <ne type="pattern">
      <ne type="hyper">metals</ne>
      such as
      <ne type="hypo">potassium</ne>
      ,
      <ne type="hypo">sodium</ne>
      ,
      <ne type="hypo">magnesium</ne>
      ,
      <ne type="hypo">iron</ne>
      ,
      <ne type="hypo">manganese</ne>
      ,
      <ne type="hypo">cobalt</ne>
      ,
      <ne type="hypo">zinc</ne>
    </ne>
    and the like, sulfates of the above-described metals, and
    phosphates of the above-described metals. More
    specifically, potassium chloride, sodium chloride,
    magnesium sulfate, ferrous sulfate, manganese sulfate,
    cobalt chloride, calcium chloride, zinc sulfate, potassium
    phosphate, sodium phosphate and the like.
  </snippet>
</P>

```

Figure 4-6: Annotated Hearst Pattern as produced by the OSCAR3 ScrapBook

The comparison of the results produced by the different annotators with one another and with the machine was carried out automatically using software built specifically for the purpose. Before two annotations can be compared, the corresponding annotations from the two sets undergoing the comparison must first be identified. This was achieved by numbering the instances of the phrase “such as” in the input texts sequentially from 0 to n , then identifying the pattern annotation, *i.e.* the `ne` element of type `pattern` which contained each instance of the phrase “such as”, if any. Each instance of the phrase “such as” may have been included in an annotation by both annotators, not included in an annotation by both annotators or included in an annotation by one annotator and not the other. Where both annotators applied an annotation, the annotations were compared on a number of criteria;

- Did the annotators apply the pattern annotation to the same section of the text?

- Did the annotators apply the hypernym annotation to the same section of text?
- How many hyponym annotations were applied by both annotators, and how many were applied by one annotator and not the other?

To answer the first two questions above, the raw text content of the appropriate annotations were stripped of all whitespace – to eliminate errors where one annotator had mistakenly included a space at the beginning or end of a word, and errors potentially introduced by the handling of whitespace in XML by the various software components involved in the execution of the task – before being compared to one another. Where the whitespace-stripped strings were equivalent it was judged that the annotators had annotated the same text and *vice versa*. To answer the third question, the sections of text to which the hyponym annotations had been applied were subjected to the same process of whitespace removal, then the sets of hyponyms annotated by each annotator were compared and the number occurring in both sets and the numbers occurring in one and not the other were calculated.

The comparison of the performance of the human annotators with that of the machine proceeded according to the same criteria but using a slightly different method. Since the machine does not create XML annotations of the form produced by the OSCAR3 ScrapBook, instead holding in memory references to the appropriate sections of a ChemicalTagger output document, it was necessary to store the value of n – the record of which instance of the phrase “such as” is the subject of the annotation – in memory as the Hearst Pattern was identified. This allowed the Hearst Patterns recognised by the machine to be aligned with those annotated by the human annotators, then the two were compared as previously.

During the analysis of the annotations, it was discovered that on a number of occasions the human annotators had produced annotations of the Hearst patterns in which the hypernym was not

marked. Such annotations were discounted from the annotation comparison procedure and the resultant metrics.

Annotator	B		C		M		Metric
A	269	45.1%	229	38.4%	133	22.3%	Annotated By Both Annotators
	234	39.2%	267	44.7%	289	48.4%	Skipped By Both Annotators
	88	14.7%	84	14.1%	174	29.1%	Skipped By One Annotator
	6		17		1		Missing Hypernym
	219	81.4%	183	79.9%	58	43.6%	Matching Patterns
	248	92.2%	206	90.0%	61	45.9%	Matching Hypernyms
	713		533		336		Matching Hyponyms
	71	9.1%	190	26.3%	17	4.8%	Mismatched Hyponyms 1
	68	8.7%	184	25.7%	45	11.8%	Mismatched Hyponyms 2
B			248	41.5%	143	24.0%	Annotated By Both Annotators
			228	38.2%	236	39.5%	Skipped By Both Annotators
			100	16.8%	218	36.5%	Skipped By One Annotator
			21		0		Missing Hypernym
			197	79.4%	59	41.3%	Matching Patterns
			216	87.1%	64	44.8%	Matching Hypernyms
			542		327		Matching Hyponyms
			171	24.0%	40	10.9%	Mismatched Hyponyms 1
			174	24.3%	69	17.4%	Mismatched Hyponyms 2
C					125	20.9%	Annotated By Both Annotators
					288	48.2%	Skipped By Both Annotators
					176	29.5%	Skipped By One Annotator
					8		Missing Hypernym
					53	42.4%	Matching Patterns
					57	45.6%	Matching Hypernyms
					243		Matching Hyponyms
					84	25.7%	Mismatched Hyponyms 1
					106	30.4%	Mismatched Hyponyms 2

Table 4-5: Results of the HearstFinder validation exercise

The results of the HearstFinder validation exercise are presented in Table 4-5, wherein the human annotators are labelled A, B and C and the machine annotator is labelled M. The metrics are calculated as follows;

- Annotated By Both Annotators – the instances of the phrase “such as” that formed part of an annotation in both annotation sets.

- Skipped By Both Annotators – the instances of the phrase “such as” that formed part of an annotation in neither annotation sets.
- Skipped By One Annotator – the instances of the phrase “such as” that formed part of an annotation in one annotation set and not the other.
- Missing Hypernym – the occasions on which a matching pair of Hearst pattern annotations were not compared as a result of one or both annotators failing to annotate the hypernym.
- Matching Patterns – the occasions on which the two annotators applied the `pattern` annotation to a whitespace-stripped equivalent section of text.
- Matching Hypernyms – the occasions on which the two annotators applied the `hyper` annotation to a whitespace-stripped equivalent section of text.
- Matching Hyponyms – the occasions on which the two annotators applied the `hypo` annotation to a whitespace-stripped equivalent section of text.
- Mismatched Hyponyms 1 – the occasions on which the first annotator applied the `hypo` annotation to a section of text which was not matched by the second annotator.
- Mismatched Hyponyms 2 – the occasions on which the second annotator applied the `hypo` annotation to a section of text which was not matched by the first annotator.

These metrics are presented as raw numbers and, where appropriate as percentages. For the metrics concerned with whether or not both annotators annotated a Hearst pattern, the percentages are calculated as a proportion of the total number of instances of the phrase “such as” in the corpus. For the pattern and hypernym metrics, the percentages are calculated as a proportion of the total number of Hearst patterns for which the comparison procedure took place, *i.e.* those

which were annotated by both annotators. For the mismatched hyponym metrics, the percentages are calculated as a proportion of the total number of hyponyms annotated by the annotator in question.

From Table 4-5, it can be seen from the inter-annotator agreement scores that the humans' interpretation and implementation of the annotation guidelines were not uniform; these scores set a baseline by which the performance of the machine may be judged. The machine performance is comparable to the inter-annotator agreement in terms of the "skipped by both annotators" metric, while the machine exhibits a higher rate, at around 30%, than the human annotators, at around 15%, in terms of the "skipped by one annotator" metric. This suggests that the machine has a tendency not to annotate Hearst patterns where the guidelines suggest that they should be annotated. This observation is to be expected as a consequence of the conservative strategy implemented by the HearstFinder. Notably, the requirement that the noun phrase that follows the key phrase "such as" be composed of terms identified as CM by OSCAR3 eliminates those Hearst patterns that contain complex hyponyms involving generic adjectives such as "higher alcohols". OSCAR3 does not recognise the word "higher" in this context and consequently the entire Hearst pattern is discounted, while a human annotator does not make this mistake.

The machine also scores significantly worse than the inter-annotator agreement in terms of the precise matching of the `pattern` and `hyper` annotations. The failure of the machine to correctly identify the hypernym can be attributed to one of two causes; the noun phrase that precedes the key phrase "such as" having been incorrectly recognised by ChemicalTagger, or the noun phrase having been correctly identified but not corresponding precisely with what the human annotator considered to be the correct hyponym. The annotation guidelines instruct the human annotators to discount from the hypernym adjectives that do not form a part of the structural class concerned, as in "suitable solvent", and the correct interpretation of this instruction requires an understanding of the English language of the chemical domain that the machine does not possess. Consequently it is

difficult to devise a system that is not prone to the second source of error, while the development of ChemicalTagger to eliminate the first source of error was beyond the scope of the current work.

The metrics for the rates of disagreement between the machine and the human annotators present a mixed picture. The inter-annotator agreement scores for mismatched hyponyms are significantly higher for the A-C comparison (26% and 26%) and the B-C comparison (24% and 24%) than for A-B (9% and 9%), and the score for the C-M (26% and 30%) comparison is significantly higher than for A-M (5% and 12%) and B-M (11% and 17%). These results are suggestive of a discrepancy in the implementation of the annotation guidelines between annotator C and annotators A and B. The scores for the comparisons A-M and B-M are similar to those of the comparison A-B, while the score for the comparison C-M are similar to those for the comparisons A-C and B-C, suggesting that the machine's performance in this regard is comparable to that of the humans. It should be remembered that these metrics are computed based solely on the Hearst patterns which have been annotated by both annotators, so although the rate at which the machine includes incorrect hyponyms may be comparable to the humans, the rate at which it excludes correct hyponyms is much higher – as evidenced by the much higher rates at which a human and the machine disagree over whether to annotate a Hearst pattern (29%, 37% and 30%) compared to the rates at which two humans disagree on the issue (15%, 14% and 17%). This conservatism is considered to be entirely acceptable, even desirable, as it assists in the production of the reliable set of relations, as seen in section 4.3.4.

4.4 Uses of Derived Data

To demonstrate the utility of the system described in this chapter, the derived relations were put to several different uses. These use cases are subsequently discussed.

4.4.1 Automatic Classification of Structural & Non-Structural Classes

In the set of derived relations, the hypernyms correspond to the names of classes of chemical compounds. Such classes may be based on structural features, *e.g.* esters, or on non-structural features, *e.g.* base. Since the compounds that form a structural class must, by definition, share structural features while those that form non-structural classes need not, it should be the case that these classes should be differentiable by considering the chemical similarity of their members.

In order to investigate this hypothesis, the set of relations produced as described in section 4.3.4 was further trimmed by removing superclasses that had fewer than six subclasses and the orphaned subclasses this process produced. The chemical similarity of each of the remaining 28 chemical classes was calculated by using the Chemistry Development Kit (CDK) version 1.0.1 to calculate fingerprints for each of the class members and the mean pairwise Tanimoto coefficient (MPT) for the class². The class names were distributed to each of five manual annotators, each of whom held at least an undergraduate degree in chemistry. Each annotator selected for each class an appropriate label selected from the following;

- Structural – the name of the class indicates that all members contain a specific substructure *e.g.* ketone or methyl ester.
- Functional – the name of the class indicates that all members share a common function, usage, property or other non-structural feature *e.g.* antibiotic or surfactant.
- Semi-structural – the name of the class indicates something about the structure or composition of the members, but not that they share a specific substructure *e.g.* isomers of C₆H₁₀O or bicyclic systems.

The annotators carried out this task by being presented with a list of the hypernyms as they were curated from the original patent texts, the order of which was randomised between the annotators,

² The CDK fingerprinter operates in a similar manner to Daylight fingerprints, by generating a comprehensive set of paths through a given connection table as opposed to by comparison to a pre-specified set of fragments. Consequently, the results are not biased by decisions made by the CDK authors regarding which fragments are appropriate for inclusion.

along with a short set of instructions. An example set of instructions and class names is included in Appendix B.

The results of this process are presented in Table 4-6, in which the chemical classes are sorted in ascending order of their mean pairwise Tanimoto (MPT) scores. Each chemical class has been assigned a label of structural, semi-structural or functional according to the consensus of the manual annotators and the classes are coloured green (structural), yellow (semi-structural) and red (functional) accordingly.

Hypernym	MPT	Manual Annotator Votes		
		Structural	Semi-structural	Functional
base	0.145			5
inert solvent	0.147			5
solvent	0.159			5
organic solvent	0.195		1	4
halogenated hydrocarbon	0.236	1	4	
mineral and carboxylic acid	0.244	3	2	
organic acid	0.256	1	3	1
hydrocarbon	0.264	1	4	
amine	0.279	5		
tertiary amine	0.288	5		
halogenated α -olefin	0.288	4	1	
aromatic ether compound	0.336	5		
cyclic olefin	0.411	4	1	
diolefin	0.431	4	1	
suitable solvent	0.448			5
trihydrocarbon-substituted phosphine	0.467	4	1	
halogenated styrene	0.499	4	1	
alkylamine	0.508	5		
olefin	0.511	4	1	
dihydrocarbon-substituted phosphine	0.527	4	1	
alcohol	0.534	5		
aliphatic unsaturated ether compound	0.546	5		
α -olefin	0.569	4	1	
aliphatic monoether compound	0.587	5		
monohydrocarbon-substituted phosphine	0.590	4	1	
alkylstyrene	0.594	5		
ether compound	0.684	4	1	
straight monoolefin	0.702	5		

Table 4-6: Classification of structural & non-structural classes

It can be seen from Table 4-6 that in 13 cases the manual annotators agreed unanimously on a label and on a further 13 occasions they agreed by a 4-1 margin. Of the remaining two cases, the class “mineral and carboxylic acid” likely indicates a union of two separate classes – “mineral acid” and “carboxylic acid” – and it is unsurprising if this caused some confusion among the annotators with regards to how to proceed – while one further class, “organic acid”, was voted semi-structural by a 3-1-1 margin.

It can be seen from Table 4-6 that ordering the classes by their MPT results in a perfect separation of the structural, semi-structural and functional classes with the exception of the classes “mineral and carboxylic acid” and “suitable solvent”. The first of these classes, as previously discussed, represents a union of two classes and was the subject of disagreement among the annotators over its nature. The second, “suitable solvent”, was unanimously voted as a functional class but has a MPT score that places it firmly among the structural classes, indicating that the process outlined in this section does not work in all cases.

The ability to automatically determine if a molecular class is structural or non-structural does not have an immediate application within the current work, but it is thought that it may prove useful in the future in several ways. Firstly, it may prove a useful pre-screen to identify structural classes for a system that attempts to automatically determine the functional groups that define them, *e.g.* esters are those compounds that contain the substructure CC(O)OC. Secondly, it may assist in the development of a system for automatic categorisation of reactions in the sense that a reaction that operates on one member of a structural class may be reasonably expected to work on other members of the class, while this is not the case for non-structural classes. It was not attempted to implement the systems described here as part of the current work, and due to the relatively low number of molecular classes examined here further work is warranted to demonstrate and validate the performance of the current methodology before attempting to build a production system.

4.4.2 Detection of Useful Relationships

It was hypothesised that the hyponymic relations identified by the application of Hearst Patterns to chemical documents may be of use for the supervised or unsupervised creation or enrichment of formal knowledge bases. In order to test this hypothesis, the set of trimmed molecular classifications identified as acceptable in section 4.3.4 were compared to the ChEBI Ontology (47; 48).

The domain of ChEBI and that of the patent corpus do not perfectly overlap. The patents cover a broad area of chemistry, while the species present in the ChEBI Ontology are described as those that are “used to intervene in the processes of living organisms (either on purpose, as for drugs, or by accident, as for chemicals in the environment)” (95). Consequently, many of the automatically acquired relations are irrelevant to ChEBI and would not be expected to be present therein. The comparison was therefore restricted to fields relevant to ChEBI. Those selected were solvents, bases and the various classes of drugs exemplified in the molecular classifications. During this work, two questions were addressed for each of the automatically identified hyponymic relations – was the chemical species represented by the hyponym present in the ChEBI Ontology and, if so, was the containing hyponymic relation defined?

The first question was addressed by searching for the chemical substance concerned both by chemical names as used in the original patents, by other common names of the substance known to the author and by the generated InChIs attached to the hyponym classes. Searching was performed using the web interface available at <http://www.ebi.ac.uk/chebi/init.do>. The second question was answered by considering whether the chemical substance was classified directly or indirectly (*i.e.* via one or more intermediate classifications, as in Figure 4-7) as belonging to a lexically or semantically equivalent classification. The results of these investigations are presented in Table 4-7, Table 4-8 and Table 4-9.

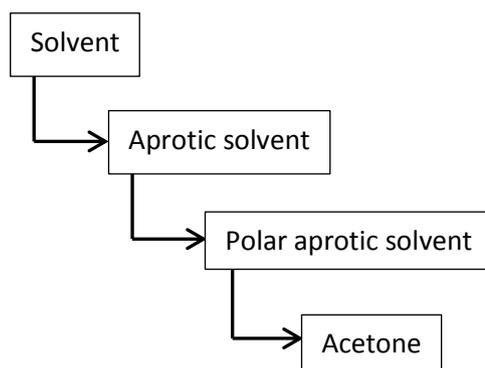


Figure 4-7: Indirect ChEBI classification of acetone as a solvent

Chemical name	Structure present in ChEBI?	Hyponymic relation present in ChEBI?
Methanol	Yes	Yes
Ethanol	Yes	Yes
DMSO	Yes	Yes
Water	Yes	Yes
Acetone	Yes	Yes
Benzene	Yes	Yes
Toluene	Yes	Yes
Glyme	Yes	Yes
1,2-dichloroethane	Yes	Yes
Chloroform	Yes	No
Xylene ³	Yes	No
Dioxane	Yes	No
NMP	Yes	No
Anisol	Yes	No
DMF	Yes	No
THF	Yes	No
Acetonitrile	Yes	No
Pyridine	Yes	No
DCM	Yes	No

Table 4-7: Comparison of solvent hyponyms with ChEBI

It can be seen from Table 4-7 that a number of common solvents, notably DMF, THF, DCM and acetonitrile, are included in the ChEBI Ontology but are not defined therein as being solvents. This is not due to solvents being considered irrelevant to ChEBI since a number of solvents are present and

³ The name “xylene” may refer to *ortho*-, *meta*- or *para*-xylene. All three were present in the ChEBI Ontology and none was recorded as being a solvent

defined as such and the ontology defines the solvent classes “polar solvent”, “non-polar solvent”, “protic solvent” and “aprotic solvent” among others. It is surprising that of the 19 solvents identified, 10 were not defined as such in ChEBI in spite of the presence of suitable superclasses.

Chemical name	Structure present in ChEBI?	Hyponymic relation present in ChEBI?
Sodium hydroxide	Yes	Yes ⁴
Potassium hydroxide	Yes	Yes ⁵
Triethylamine	Yes	No
Pyridine	Yes	No
n-BuLi	Yes	No
Imidazole	Yes	No
Potassium carbonate	No	N/A
Sodium hydride	No	N/A
N,N-diisopropyl-N-ethylamine	No	N/A
DMAP	No	N/A

Table 4-8: Comparison of base hyponyms with ChEBI

Table 4-8 demonstrates that the formal definition of bases in ChEBI is poor. Indeed, a manual search indicates that the “metallic base” class to which sodium and potassium hydroxide belong is the only class of bases defined.

⁴ Recorded in ChEBI as a “metallic base”

⁵ Recorded in ChEBI as a “metallic base”

Drug class	Chemical name	Structure present in ChEBI?	Hyponymic relation present in ChEBI?
Biguanide	Metformin	Yes	Yes
Cholesterol absorption inhibitor	Ezetimibe	Yes	Yes ⁶
Anti-estrogen	Tamoxifen	Yes	Yes ⁷
	Toremifene	Yes	Yes ⁸
	Raloxifene	Yes	Yes ⁹
H ⁺ , K ⁺ ATPase inhibitor	Omeprazole	Yes	No
	Lansaprazole	Yes	Yes ¹⁰
Antibiotic	Tetracycline	Yes	Yes
	Metronidazole	Yes	No ¹¹
	Amoxicillin	Yes	Yes
Tricyclic ¹²	Amitriptyline	Yes	Yes
	Imipramine	Yes	Yes
	Doxepin	Yes	Yes
Antipsychotic drug	Thoridazine	Yes	Yes
	Haloperidol	Yes	Yes
Psychostimulant	Methylphenidate	Yes	No
Diuretic	Amiloride	Yes	No
	Furosemide	Yes	Yes

Table 4-9: Comparison of drug hyponyms with ChEBI

It should be expected that the drugs included in the molecular classifications should all be present in the ChEBI Ontology, since in order for a hyponymic relation and therefore its hyponym to be included, the hyponym must be resolvable to a connection table by OSCAR3 – which uses ChEBI as a source of linked trivial names and associated structures. It might also therefore be expected that the

⁶ Recorded in ChEBI as a “anticholesteremic drug”

⁷ Recorded in ChEBI as an “estrogen receptor antagonist”

⁸ Recorded in ChEBI as an “estrogen antagonist”

⁹ Recorded in ChEBI as an “estrogen receptor modulator”

¹⁰ Recorded in ChEBI as a “proton pump inhibitor”

¹¹ Recorded in ChEBI as a “antitrichomonal drug”

¹² A common abbreviation of “tricyclic antidepressant”, of which class the three examples are recorded as members in ChEBI

drug classifications deduced from the literature would similarly be formally encoded into ChEBI, though this is not the case for 4 of the 18 drugs present in the corrected set of molecular classifications.

This observation, as well as the high proportion of identified solvents and bases that were not present in or defined as such in ChEBI suggest that the technology developed for the automatic acquisition of hyponymic relations is likely to be of use in the creation of such formal classifications of chemical substances. Though it is likely that a human curator would not want to enable fully-automatic deposition of hyponymic relations due to justifiable concerns about accuracy, such a system may act as a highly useful tool to identify both common molecular classes and examples thereof.

4.4.3 Application to Data Searching

The automatic generation of dictionaries of common molecular classes provides a means of support for data searching in that it permits the formulation of queries such as “show me reactions that use a strong base in a chlorinated solvent” without the need for the user to define the terms “strong base” or “chlorinated solvent”. The combination of the hyponymic relations derived in the current work with data derived from other sources is made possible by the use of the web standard OWL for their formal descriptions. This work has been carried out by Dr Lezan Hawizy and is discussed in section 6.2.

4.5 Conclusions

The work in this section has demonstrated the implementation of a system capable of the automatic detection of molecular classes based on their specification in the literature that operates with a high

degree of accuracy by combining established methods with novel technology. It has further discussed ways in which the information produced may be of use to the human curators of the ChEBI ontology and to the broader community. The techniques used in this process are ones that scale acceptably with the available volume of literature and it is believed that they will be of value in the future.

5. High-Throughput Abstraction of Chemical Reactions – PatentEye

This chapter describes the implementation of a novel framework – tentatively named PatentEye – for the high-throughput and automatic abstraction of chemical reactions. As discussed in the introduction to the current work, the liberation of scientific data and its conversion to machine-understandable forms holds great promise. A key part of the chemical sciences are the reactions that chemists perform and report in great number, and the goal of the creation of PatentEye was to demonstrate the potential to create an automated system capable of extracting reactions from the literature, creating machine-understandable representations using Chemical Markup Language (CML) and sharing them as open data. To increase the reliability of the extracted syntheses, PatentEye attempts to validate the identified product molecules. This is achieved by comparison of a candidate product molecule with any accompanying structure diagram using the package OSRA (see section 2.2.8) for image interpretation and with any accompanying NMR and mass spectra, using the OSCAR3 data recognition functionality (see section 2.2.6.5). The identified NMR spectra are considered to be valuable data in their own right and are extracted and retained for use in later works.

The implemented system is automated to the degree that it is capable of operating with minimal user interaction, and consequently the PatentEye workflow consists of a number of stages of processing. First, chemical patents are identified within the online archive of the European Patent Office (EPO) and are downloaded. The XML documents supplied by the EPO are then semantically enhanced so as to delimit sections and subsections of the text and to introduce additional metadata such as SMILES strings representing the content of structure diagrams and OSCAR3 data markup to describe identified spectra. Finally, reactions are extracted from these semantically enhanced documents using ChemicalTagger (see section 2.2.7) and are converted to CML. The working of each of these steps is described in this chapter.

5.1 Downloading Patents

As discussed in section 2.1.3, the patents published by the European Patent Office were selected for the current work. In order to simplify the acquisition of a suitably large corpus of documents from which to work, as well as to facilitate future increases in the scale of PatentEye's operation, software was written to automate the process of identification and downloading of the patent files. The operation of this software is subsequently discussed.

5.1.1 EPO Web Interface

The European Patent Office (EPO) publishes patent documents through the European Publication Server, hosted at <https://data.epo.org/publication-server/>. This platform is designed for a human, interacting with it using a web browser. A user performs a search by entering his search parameters – a patent ID, a date range within which to search and a list of document kinds (see Table 5-1) to search for – into an HTML form, and upon submission an HTTP POST request to <https://data.epo.org/publication-server/search> is executed, specifying the required parameters. The server responds by redirecting the browser to <https://data.epo.org/publication-server/result-list> using the HTTP 302 status code, and the page at this address uses Asynchronous Java and XML (AJAX) to retrieve a table of results for the search, an example of which is shown in Figure 5-1.

	Publication number ¹	Kind code	Publication date	XML ¹	PDF/PCT	ZIP ¹		
<input type="checkbox"/>	EP 1777210	A1	2007/04/25	XML	PDF	ZIP	esp@cenet	Register
<input type="checkbox"/>	EP 1777210	B1	2009/05/27	XML	PDF	ZIP	esp@cenet	Register

Figure 5-1: Search results for EP 1777210

This table comprises a list of the available documents related to the specified patent. The documents are each assigned a “kind code”, which classifies the contents of the documents according to the definitions shown in Table 5-1.

Kind code	Definition
A1	European patent application published with European search report
A2	European patent application published without European search report
A3	Separate publication of the European search report
A8	Corrected title page of an A document, <i>i.e.</i> A1 or A2 document
A9	Complete reprint of an A document, <i>i.e.</i> A1, A2 or A3 document
B1	European patent specification (granted patent)
B2	New European patent specification (amended specification)
B3	European patent specification (after limitation procedure)
B8	Corrected title page of a B document, <i>i.e.</i> B1 or B2 document
B9	Complete reprint of a B document, <i>i.e.</i> B1 or B2 document

Table 5-1: Definitions of EPO kind codes, taken from EPO website (96)

The search results table also provides links to the XML, PDF and ZIP files for the document, if these are available. Not all file types are available for all patent documents – if the patent is published under the Patent Cooperation Treaty (PCT), for example, then the PDF link is replaced by a PCT link and a full-text XML version is unavailable.

5.1.2 Automated Downloading of EPO patents

In order to download documents using the web interface described previously, it is first necessary to determine an appropriate patent ID number. A user may already know the ID of the patent in which he is interested if, for example, it has been cited by another document or if it has been returned in the hit list of a patent searching tool. The European Publication Server, in addition to hosting the search interface, also publishes a series of weekly patent index files (97). Each of these files lists the patent documents published by the EPO in a certain week and provides further information such as the language in which the document is written and the International Patent Classification (IPC) codes

which have been assigned to the document. The IPC is a subject-based hierarchical classification scheme for patent documents, *e.g.* chemistry and metallurgy are assigned the code “C”, organic chemistry is assigned the code “C07”, acyclic or carbocyclic compounds are assigned the code “C07D” *etc.* These index files thereby provide the means for an automated system such as PatentEye to identify chemical patents, or indeed those related to any other field. PatentEye identifies patents to download as those that are written in English and that have been assigned one or more of the IPC codes listed in Table 5-2.

IPC Code	Description
C07B	General methods of organic chemistry; apparatus therefor
C07C	Acyclic or carbocyclic compounds
C07D	Heterocyclic compounds
C07F	Acyclic, carbocyclic or heterocyclic compounds containing elements other than carbon, hydrogen, halogen, oxygen, nitrogen, sulfur, selenium or tellurium

Table 5-2: Relevant IPC codes

The implementation of this process in PatentEye is provided by the `EpoCrawler` class. Once the list of chemical patent IDs has been derived from a patent index file, the IDs are passed to the `PatentGrabber` class, which uses the web crawler developed for the CrystalEye project (13) and interacts with the patent-searching web interface described above. The search request is submitted, and then the table of results is retrieved by replicating the AJAX request. This table is formatted in XHTML, allowing the `PatentGrabber` to identify which kinds of documents are available for the specified patent by reading the individual table rows. For the documents of kind A1, A2, A9 and B1 – each of which contains the full text of a patent – the URLs from which the ZIPs may be downloaded are identified and passed to the web crawler for download.

5.1.3 Formation of the Patent Corpus

The EpoCrawler was used to download the zipped form of the chemical patents from the EPO website for the ten weeks dated from May 6th 2009 to July 8th 2009. To prevent duplication of patents within the corpus, files were then deleted such that only one document remained within the corpus for each patent ID. The order of priority used to determine which document to retain was A9 > A1 > A2 > B1. In total, the patent corpus comprises 690 documents from across the ten weeks, and the number of ZIP files originally downloaded and the number of unique patents from which they are drawn are shown in Figure 5-2.

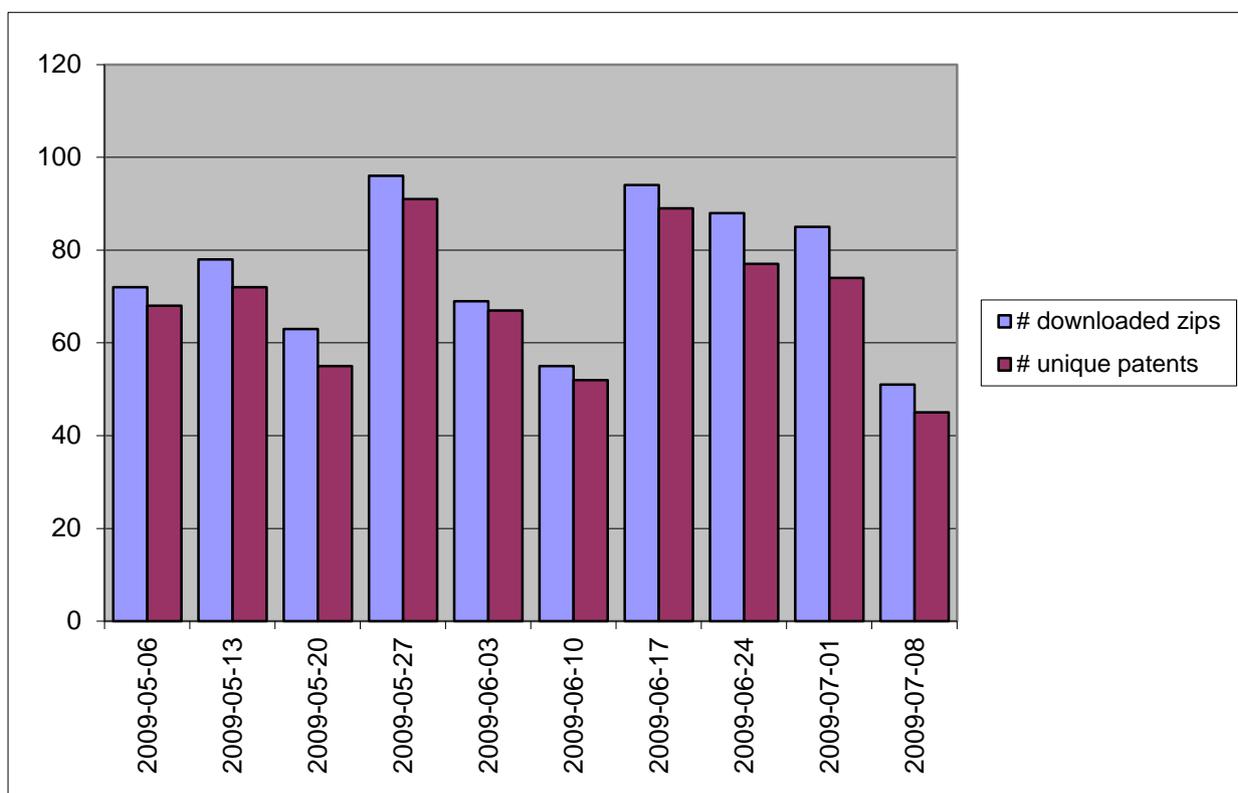


Figure 5-2: Variation of downloaded and unique patents in the corpus

Of these 690 zips, it was found that 23 did not contain the XML version of the patent under the expected file name. Further work that uses the patent corpus is therefore based on a reduced corpus of 667 unique, full-text patent documents where the XML files are used as input.

5.2 Document Enhancement

As discussed previously, in section 2.1.3.1, the different sections of the XML-formatted patent documents are not clearly defined. The content of the `description` element is relatively flat – that is to say, the `heading` and `p` (paragraph) children are siblings of one another, such as in the following;

```
<description>
  <p>...</p>
  <heading>Heading 1</heading>
  <p>...</p>
  <p>...</p>
  <heading>Heading 1.1</heading>
  <p>...</p>
  <heading>Heading 1.2</heading>
  <p>...</p>
  <heading>Heading 2</heading>
  <p>...</p>
  <p>...</p>
</description>
```

To a human reader, it is a simple task to realise that the headings 1.1 and 1.2 are subsections of Heading 1, and that each of the paragraphs belongs to a section of the document that begins with the preceding heading. Since this is not made explicit in the structure of the XML, however, it is not trivially obvious to a machine that the document should be read in such a way. For this reason it is desirable to deflatten the XML – to rewrite the document such that as much of the implicit structure is made explicit as possible. This rewritten document is then saved to disk in order to prevent unnecessary repetition of the task.

A number of other semantic enhancements are performed on the patent documents at this stage. These tasks include the application of OSCAR3 data recognition to identify spectral data within the text, the application of OSRA to add SMILES representations of the chemical structure images contained within the documents, the recognition and annotation of references in the text to other sections of the document, *e.g.* “the reaction was performed as in example 12” and the identification

and labelling of the paragraphs in the text that form part of an experimental section. The theory and application of these steps are subsequently discussed.

5.2.1 Paragraph Deflattening

In this step, the `description` element of the patent document is checked for paragraph children. Any `p` elements that are found are detached from the document and re-attached as a child of the heading element that most recently precedes them. Any `p` elements that occur before the first heading child of the `description` element are ignored by this process. For example, the example of XML in the preceding section would be reformatted as follows;

```
<description>
  <p>...</p>
  <heading>Heading 1
    <p>...</p>
    <p>...</p>
  </heading>
  <heading>Heading 1.1
    <p>...</p>
  </heading>
  <heading>Heading 1.2
    <p>...</p>
  </heading>
  <heading>Heading 2
    <p>...</p>
    <p>...</p>
  </heading>
</description>
```

Before this reformatting, the `heading` element was acting as an annotation on the heading text. While it can still be inferred that the text inside a `heading` element and preceding the first `p` element is the heading text, the reformatting process has destroyed the explicit declaration and created mixed content. To remove the requirement to infer the heading title, the heading text is removed from the document and made into a `title` attribute on the `heading` element, to form a document of the following form;

```
<description>
  <p>...</p>
  <heading title='Heading 1'>
    <p>...</p>
    <p>...</p>
  </heading>
  <heading title='Heading 1.1'>
    <p>...</p>
  </heading>
  <heading title='Heading 1.2'>
    <p>...</p>
  </heading>
  <heading title='Heading 2'>
    <p>...</p>
    <p>...</p>
  </heading>
</description>
```

5.2.2 Document Segmentation

As previously discussed, the EPO do not attempt to explicitly demarcate in their XML the existence of sections of a patent document. Headings in the document are denoted by use of the `heading` tag, but otherwise the reader is left to infer for themselves where subheadings occur and to which headings they belong. This lack of formal structure in the document is a barrier to the automated processing of the patent documents as it prevents a machine from making context-specific decisions about how to behave. At this stage in the semantic enrichment process, an attempt is made to formalise the document's implicit structure.

5.2.2.1 Primary Sections

The EPO's instruction document, "How to get a European Patent – Guide for Applicants" (98), states;

In the description you must:

- (a) Specify the technical field to which the invention relates. You may do this for example by reproducing the first ("prior art") portion of the independent claims in full or in substance or by simply referring to it.
- (b) Indicate the background art of which you are aware, to the extent that it is useful for understanding the invention, preferably citing source documents reflecting such

art. This applies in particular to the background art corresponding to the prior art portion of the independent claims. Source document citations must be sufficiently complete to be verifiable: patent specifications by country and number; books by author, title, publisher, edition, place and year of publication and page numbers; periodicals by title, year, issue and page numbers.

- (c) Disclose the invention as claimed.
- (d) The disclosure must indicate the technical problem that the invention is designed to solve (even if it does not state it expressly) and describe the solution.
- (e) To elucidate the nature of the solution according to the independent claims you can repeat or refer to the characterising portion of the independent claims (see example) or reproduce the substance of the features of the solution according to the relevant claims.
- (f) At this point in the description you need only give details of embodiments of the invention according to the dependent claims if you do not do so when describing ways of performing the claimed invention or describing what the drawings show.
- (g) You should state any advantageous effects your invention has compared with the prior art, but without making disparaging remarks about any specific previous product or process.
- (h) Briefly describe what is illustrated in any drawings, making sure you give their numbers.
- (i) Describe in detail at least one way of carrying out the claimed invention, typically using examples and referring to any drawings and the reference signs used in them.
- (j) Indicate how the invention is susceptible of industrial application within the meaning of Article 57.

Each of these six points defines a topic that must be addressed in the patent. Consequently, European patents tend to follow a regular structure – primary sections of the documents are commonly headed according to the areas mandated in the instructions using relatively standardised terms. As a result, these heading titles may be matched using regular expressions. The regular expressions used for matching the most common section titles are given in Table 5-3. Each of these section titles corresponds to one of the areas mandated by the guide for applicants, with the exception of the final entry – while a summary of the invention is not mandated by the EPO it is a very common feature, and one for which the terminology is sufficiently consistent that it can be matched by regular expressions.

Section Title	Regular Expression
Field	<code>(.*\s+)?(technical\s+)?field(\b.*)?</code>
Prior Art	<code>(.*\s+)?(prior background related)(\b.*(art(s)? invention)(\b.*)?)?</code>
Disclosure of Invention	<code>(.*\s+)?(detailed description (disclosure description)\b.*\s+invention)\b.*</code>
Description of Drawings	<code>(.*\s+)?description of (.*\s+)?drawing(s)?(\b.*)?</code>
Mode of Carrying Out	<code>(.*\s+)?modes?(\s+.*carry.+(\s+.*)?)?</code>
Industrial Applicability	industrial applicability
Summary of Invention	<code>(.*\s+)?summary(\s+.*invention(\b.*)?)?</code>

Table 5-3: Regular expressions for identifying primary section headings

These regular expressions are used to identify the primary sections of the patent description. Once this has occurred, the structure of the descendant elements of the patent's `description` element is modified with the intent that the only child elements of the `description` should be the identified primary section headings. Non-primary headings that are found after a primary heading are detached from the document and reattached as a child element of the preceding primary heading, leaving those that occur before the first identified primary heading in place. In this manner, the flat structure of the XML provided by the EPO is converted into a tree that reflects the implicit structure of the document.

In addition to this alteration of the XML structure, the names of the primary heading elements are normalised to enable their trivial location within the document during later work. Each of the primary headings has a separate keyword to which the element name is changed, for example the disclosure of invention headings are renamed "disclosureOfInvention", while the summary of invention headings are renamed "summaryOfInvention". Similarly, the remaining `heading` elements within the `description` that match the regular expression `(.*\b)?example(:|\-)?(\s+.*)?` are renamed "example" in the same manner as the primary section headings.

Though the PatentEye functionality implemented as part of the current work does not ultimately make use of the primary section headings, the ability to identify the primary sections is thought to be of sufficient potential use to future applications that it has been retained within the current codebase. Since the functionality was not used in the current work, the accuracy of the regular expressions at recognising variations in the wording of the primary section headings by individual authors was not assessed.

5.2.2.2 Identification of Consecutive Headings

Very often, a document contains a set of headings that are intended to form a series of consecutive headings. Most notably in the field of patents, it is clearly apparent that, if a document contains the headings “example 1”, “example 2” and “example 3” that these headings form a series and that, in general, other headings that occur during such a series are in fact subheadings of the those headings that form the series. This phenomenon provides a useful means by which a machine may identify the implicit structure within the document, and by re-ordering the semantics of the document it is possible to make this structure explicit. This procedure is implemented as part of the current work and is subsequently discussed.

Recognisable Consecutive Headings

Two formats of consecutive headings were identified for this work: those where the heading text is invariant save for an incrementing index *e.g.* “example 1”, “example 2”, *etc.* and those in which the headings do not necessarily share any text, but the sequential nature of the headings is identifiable from the presence of the incrementing token at the beginning of the headings, *e.g.* “A. Introduction”, “B. Methods”, *etc.* It was also noted that in headings describing example compounds the name of the compound it is common to include the name of the compound, *e.g.* “Step 2:

Preparation of methyl 5,8-dihydroxy-1,6-naphthyridine-7-carboxylate”, and that this should not be allowed to prevent the formation of lists of consecutive headings.

In order to recognise these sets of consecutive headings, a representation is first generated of the title text of each of the headings in which the incrementable tokens and chemical names have been normalised. Firstly, chemical names are identified by passing the title text to OSCAR3 for named entity recognition and by replacing instances of chemical names with the string “\$CM”. Subsequently, incrementable tokens are identified by matching against the regular expression;

```
(\\d+[a-z]? (?!-) \\b|^ [A-Z] (?!-) \\b)
```

This regular expression matches two types of incrementables. Firstly, it may match a number optionally followed by a single letter (*e.g.* “7” or “12b”) followed by a word boundary and not a dash (to avoid matching pieces of chemical nomenclature such as in “2-methyl”). Secondly, it may match a single uppercase character at the beginning of a string that is, again, followed by a word boundary but not a dash (“N-methyl”). The substrings that are matched by this regular expression are replaced with the string “\$IN”. Representations created in this way may then be compared by simple string equivalence to determine if the headings for which they were generated are of a common format. If so, and if the incrementing tokens are consecutive, then the headings may constitute consecutive headings.

The application of this procedure needs to be carefully applied in order to correctly identify lists of consecutive headings. Consider the following list of headings;

Example 1
Step 1
Step 2
Example 2
Step 1
Step 2
Step 3

It should be clear to the reader that these headings describe two examples, each of which is broken down into a number of steps. The first example has two subheadings while the second has three. The heading “Step 2” in “Example 1” has no subsequent heading in its consecutive heading list, and it is important not to identify “Step 3” in “Example 2” as such. This is achieved by dividing the list of headings in the document into smaller lists after each identification of consecutive headings, using the headings in the consecutive heading list as the splitting points. This procedure is illustrated in Figure 5-3.

Example 1	-> Example \$IN	-> Example \$IN	-> Example \$IN
Step 1	-> Step \$IN	-> Step \$IN	-> Step \$IN
Method	-> Method	-> Method	-> Method
Characterisation	-> Characterisation	-> Characterisation	-> Characterisation
Step 2	-> Step \$IN	-> Step \$IN	-> Step \$IN
Method	-> Method	-> Method	-> Method
Characterisation	-> Characterisation	-> Characterisation	-> Characterisation
Example 2	-> Example \$IN	-> Example \$IN	-> Example \$IN
Step 1	-> Step \$IN	-> Step \$IN	-> Step \$IN
Method	-> Method	-> Method	-> Method
Characterisation	-> Characterisation	-> Characterisation	-> Characterisation
Step 2	-> Step \$IN	-> Step \$IN	-> Step \$IN
Method	-> Method	-> Method	-> Method
Characterisation	-> Characterisation	-> Characterisation	-> Characterisation
Step 3	-> Step \$IN	-> Step \$IN	-> Step \$IN
Example 3	-> Example \$IN	-> Example \$IN	-> Example \$IN

Figure 5-3: Identification of and Document Restructuring Using Consecutive Headings

In the first step, the title text of each of the headings is normalised by replacement of chemical names and incrementable elements, as previously discussed. The first heading, “Example 1”, is then identified as being part of a consecutive heading list with “Example 2” and so the list of headings is divided into two, shown in blue and green, using these headings as the splitting points. Now, when it comes to find the headings that follow the blue “Step 1” heading, it is impossible to include any of the subheadings of “Example 2”, as these headings are no longer contained in the heading list. This results in the correct identification of lists of step headings, as shown in the example above.

Once the lists of consecutive headings have been created, the document is restructured accordingly. Firstly, a check is carried out to ensure that the headings in each list are siblings, *i.e.* that they share a common parent element. If this is found not to be the case, the process aborts by throwing an `UnrecognisedStructureException`. Otherwise, the document restructuring proceeds by iterating through the list of all headings that are siblings of those in the list. Once the first heading in the list is reached, subsequent headings that are not also members of the list are detached from the document and reattached as children of the preceding member. This process is terminated upon reaching the final member of the list, with the result that the subheadings of this heading are not affected. This omission is accepted as a consequence of the fact that there is not another method available to determine its subheadings.

5.2.3 Data Annotation

PatentEye takes advantage of the previously-developed and previously-described functionality for the recognition and annotation of experimental data that exists within OSCAR3. To enable the usage of this functionality, software was developed to support the application of OSCAR3 data annotations to an original source XML document and improvements were made to the performance of the OSCAR3 data recognition. This work is subsequently described.

5.2.3.1 Development of Annotation Framework

OSCAR3 is used to annotate reports of spectral data within the patent documents. OSCAR3, however, does not directly allow for the annotation of arbitrary XML documents. Instead, the patent documents must first be converted to SciXML, as described previously, before annotation is

performed. This produces an annotated SciXML document, but the process of format conversion has destroyed much of the valuable markup that is included in the patent XML files.

To work around this problem, software was written to identify the sections of text in the original document that correspond to the sections annotated by OSCAR3 in its SciXML documents in order to allow the annotations to be added to the original document. It is hoped that this process will be made redundant by the addition of the capability to annotate arbitrary XML documents in a future version of the OSCAR software.

The process of creating and transferring OSCAR3's data annotations is illustrated in Figure 5-4.

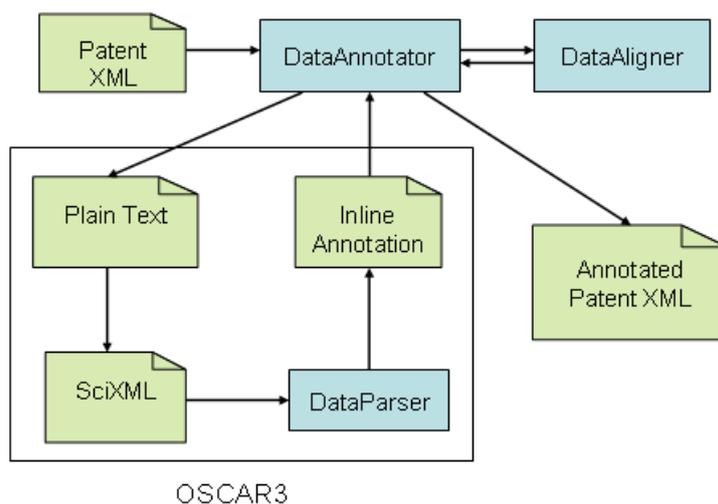


Figure 5-4: Software architecture for the application of OSCAR3 data annotations to patent XML documents

The `DataAnnotator` class operates on a single paragraph of text, held in memory as a `nu.xom.Element`. The text content of the paragraph is handed to OSCAR3 to create a SciXML document, and this document is used by the `DataParser` class in OSCAR3 to produce a set of inline annotations. A user-modifiable subset of the inline annotations, by default comprising those indicating the presence of NMR, mass spectrometry, high-resolution mass spectrometry and IR spectroscopy, are then identified. The `DataAligner` class then locates the equivalent, unannotated, sections in the XML source of the input paragraph. This process does not proceed by

simple substring matching, since the process of conversion to SciXML has normalised whitespace and removed markup from the source. Consider the example of the following input paragraph;

```
<p>
  The compound was characterised as follows; MS <i>m/z</i>
  189(M<sup>+</sup>). This confirms the experimental product as
  the desired product.
</p>
```

In the SciXML document on which OSCAR3 works, the `i` and `sup` tags will have been removed, and so they will be absent from the annotation produced by OSCAR3. The source XML of the input paragraph and the annotation, however, both share the text “MS m/z 189(M+)”. The `DataAligner` class, where possible, identifies the substring of the source XML that contains the text value of the annotation and, where present, XML tags from a predetermined set that correspond to style markup (such as in the example above) or surplus whitespace. The data annotation that has been created by OSCAR3 may then be copied into the input paragraph by replacing the section of source XML matched by the `DataAligner` with the source XML for the annotation. If this process cannot complete successfully, the `DataAnnotator` throws a `FailedInlineException` to indicate failure.

It is important to realise that the process described as above cannot be guaranteed to produce well-formed XML. Consider the following input paragraph;

```
<p>
  <sup>1</sup>H NMR (400MHz): δ 1.20 (2H, s), 2.34 (1H, t,
  J=2.3Hz), 3.78 (2H, d, J=2.3Hz).
</p>
```

In this case, the substring identified as a match by the `DataAligner` will include the closing `</sup>` tag but not the opening `<sup>` tag. When this substring is replaced by the source XML for the annotation, this will result in the presence of an unbalanced opening tag. As a result, before the source XML produced by the method outlined above can be built to produce a paragraph suitable

for insertion into the original document, it is necessary to identify and remove any such unbalanced tags. This process is handled by the DataAnnotator.

5.2.3.2 Measurement of OSCAR3 Performance

In order to test the performance of OSCAR3's DataParser on the EPO patents, a corpus of paragraphs was constructed from those patents that had successfully passed through the paragraph deflating and document segmentation phases of the semantic enrichment procedure. These paragraphs were selected at random from those that were descendants of an `example` element, with each paragraph having an equal probability of selection. It was decided to limit the domain in this way in order to enrich the proportion of paragraphs containing experimental data and thereby reduce the time required to produce an annotated corpus of appropriate size. Of the paragraphs selected in this way, 1400 were divided into two sets of 700 each – one for development of the OSCAR3 regular expressions and one for validation of the development process. These paragraphs were manually examined, and sections of experimental text within the corpora were annotated according to the guidelines in Appendix C. The manual annotation of the documents was intended to show the position of the spectral data within the text and the type of spectrum present, but not to fully identify the components of the spectra *e.g.* peaks, shifts, *etc.* As a result, the annotated paragraphs were of the form;

```

<p id="p0166" num="0166" patent="EP 1343782B1.xml">
  Intermediate Example 17 (7 g, .037 mol) and 10% Pd/C (.7g)
  in a concentrated methanol solution were shaken under
  approximately 40 psi of H<sub>2</sub> in appropriate
  pressure vessel using a Parr Hydrogenator. When the reaction
  was judged to be complete based upon the consumption of the
  nitrobenzimidazole, it was diluted with EtOAc and filtered
  through Celite and silica gel, which was washed with a
  mixture of EtOAc and MeOH and concentrated. The product was
  carried on without purification.
  <spectrum type='hnmr'>
    1H NMR (300 MHz, d<sub>6</sub> DMSO) δ 7.11 (d, J = 8.38
    Hz, 1H), 6.69 (d, J = 1.51 Hz, 1H), 6.53 (dd, J = 8.38,
    1.51 Hz, 1H), 4.65 (s, 2H), 3.62 (s, 3H), 2.43 (s, 3H)
  </spectrum>
  .
</p>

```

Accuracy of the OSCAR3 data annotations is assessed by the `OscarValidator` class. Each manually annotated paragraph is read into memory as a `XOM Element` and an un-annotated copy made by removing the `spectrum` tags and rebuilding the XML as a `XOM Element`. This un-annotated paragraph is then passed to a `DataAnnotator` to produce an automatically-annotated paragraph that can be compared against the original manually-annotated copy. This comparison is performed by string comparison of the text of the annotations in the manually and automatically annotated versions of the paragraph and by comparing the values of the `type` attribute of the annotation. If an annotation of the same type with the same text is found in both the automatic and manually annotated paragraphs, a true positive (TP) is recorded. If an annotation from the manually-annotated paragraph is not matched in this way in the automatically-annotated paragraph, a false negative is recorded, and if an annotation from the automatically-annotated paragraph is not matched by one in the manually-annotated paragraph, a false positive is recorded. Validation of the version of the `DataParser` contained in the OSCAR3 α5 release (99), the version that existed prior to the commencement of this work, using the development paragraphs produced the results shown in Table 5-4, where precision measures the proportion of identified reagents that were correctly identified by the machine and recall measures the proportion of reagents specified in the text that were correctly identified by the machine. The two measures are defined as follows;

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

Spectrum type	# in corpus	TP	FP	FN	Precision	Recall
MassSpec	227	104	51	123	67.1%	45.8%
HNMR	206	97	21	109	82.2%	47.1%
CNMR	12	6	2	6	75.0%	50.0%
IR	10	0	0	10		0.0%
HRMS	5	0	0	5		0.0%

Table 5-4: Performance of OSCAR3 α 5 on the development corpus

The low rate of occurrence of CNMR, IR and HRMS data make it difficult to draw conclusions from the performance of the data annotation, but the rates of recall for the MassSpec and HNMR data are disappointingly low compared to previously published values (100). This is due in part to the origin of the data recognition as part of the Experimental Data Checker application. The purpose of this application was to highlight mistakes made by authors in preparing their manuscripts to RSC requirements. It is desirable in that context to fail to identify data sections that do not conform to the style guidelines of the journal concerned in order to highlight the author's mistake, but in the context of the current task it is appropriate to loosen this strict requirement in order to increase rates of recall.

5.2.3.3 Development of OSCAR3 Data Recognition

By manual comparison of the human and OSCAR3 annotations, a number of classes of common errors were identified. These errors are subsequently discussed. Illustrative examples are taken from the corpus of paragraphs that were examined during the development process.

Unspecified NMR Type

OSCAR3 recognises both ^1H and ^{13}C NMR and distinguishes between the two by requiring the presence of the literal string “ ^1H ” or “ ^{13}C ” respectively near the beginning of the text to be matched. Many of the reported NMR from the patents omitted this declaration, instead taking the form “NMR(CDCl_3): δ 0.88(d, J = 6.6 Hz, 6H)...”. A trained chemist will easily infer that this as a ^1H NMR since the integral is specified as “6H”, however this form was not matched by any of the OSCAR3 regexes and it therefore went unannotated. This problem was fixed by the creation of a new type of spectrum to be annotated – the unknownNmr. A spectrum is annotated as unknownNmr only if the isotope under examination is not identified, and no attempt is made to infer it from the text. Furthermore, it was noticed that some NMR spectra specify simply the element rather than the isotope under investigation, and were not being matched by the regular expressions. This problem was resolved by making the “1” and “13” preceding “H” and “C”, respectively, optional.

Inclusion of “ppm”

This problem occurred when the author appended “ppm” to the end of a spectrum, *e.g.* “ ^{13}C -NMR (CDCl_3): 12.02 (CH_3), 23.64 (2C), 32.28 (2C), 38.81 (2C), 49.64, 54.09 (CH), 211.83 (C=O) ppm”. In

these cases, OSCAR3 would annotate the spectrum as far as the final peak but omit to include the “ppm” in the annotation. This problem was trivially fixed by allowing the inclusion of this final token.

Negative Shifts

While uncommon, negative values of NMR shift are entirely valid, but were not accepted as such by OSCAR3. Upon examination, this was discovered to be part of a wider bug in the data regexes – both the character “+” and “-” were intended to be allowed before numbers to indicate sign. In regular expressions, however, the hyphen is a metacharacter when it occurs within a character class and must therefore be escaped to give a literal hyphen in these circumstances. In a number of occasions throughout the data regexes unintentional unescaped hyphens were used inside character classes, and where identified this problem was resolved.

Word Boundaries around the ‘Delta’ Character

It was noted that OSCAR3 was failing to annotate NMR spectra which contained the substring “δ:”, *e.g.* “¹H-NMR (CDCl₃) δ: 2.34 (3H, s), 2.66 (3H, s), 8.12 (1H, s)”. The top-level regular expression that captures a ¹H NMR spectrum in OSCAR3 α5 is as follows;

```

1 <node type="spectrum" id="hnmr" value="hnmr">
2   <regexp parsegroup="0">
3     <insert idref="nmrDelta" />?
4     \b
5     <insert idref="hNmr.Prolog"/>
6     (?: \W* for\s+\w+ (?: (![\(\);])*)*)?
7     <insert idref="nmrMethod"/>?
8     (\W+(<insert idref="nmrDelta"/>|H)+\b)?
9     [\s:=]+?
10    (?: \W*ppm\W*)?
11    (?: peaks\s+at\s+)?
12    (?:\s*<insert idref="nmrDelta"/>\s+)?
13    <insert idref="nmrPeakBlock"/>
14  </regexp>
15  <child type="quantity" id="hnmrSolvent"/>
16  <child type="quantity" id="hnmrStandard"/>
17  <child type="quantity" id="hnmrFrequency"/>
18  <child type="quantity" id="hnmrTemperature"/>
19  <child type="peaks" id="hnmr"/>
20 </node>

```

As can be seen above, the NMR Delta is matched in one (or more, if necessary) of three places – on lines 3, 8 and 12. Following the first two of these is non-optional word boundary (“\b”). A word boundary in regular expressions is a zero-length match that can be made at the transition between a word character and a non-word character, a word character being any one of the lower case letters a-z, the upper case letters A-Z, the digits 0-9 and the underscore, “_”. The NMR Delta is subsequently defined as;

```
<def id="nmrDelta" type="const">(?:d|đ|δ|ä)</def>
```

When the NMR Delta is represented by the character “d” it is a word character, and when it is represented by the character “δ”, or other it is not. Thus, the NMR Delta is followed by a word boundary if the next character is a non-word character if the NMR Delta is a “d” and *vice versa* otherwise. The NMR Delta is rarely immediately followed by a word character, thus the literal delta, δ, is generally only matched on line 12 of the preceding regular expression. This is problematic where the delta is followed by one of the characters “=” and “:”, since these are only matched on line 9, *i.e.* before the literal delta can be matched. This problem was resolved by removing the

requirement on line 8 for the NMR Delta to be followed by a word boundary, allowing for the character “ δ ” to be matched at this point.

Typos

It was observed that a number of the manually annotated spectra were not recognised by OSCAR3 simply because they contained typos. Missing commas in lists of peaks, missing close brackets following a peak assignment and inappropriately positioned spaces, *e.g.* in the middle of a number were observed, along with less frequent typos, and caused OSCAR3 to fail to annotate the spectrum correctly. Frequently, these mistakes would also cause OSCAR3 to incorrectly annotate a subsection of the full spectrum, producing a false positive in addition to the false negative. For example;

False negative: 1H-NMR(CDCl₃,TMS) δ (ppm):2.55(3H,s),6.75-6.85(1H,m),7 .03(2H,d,J=8.4Hz),7.1-7.2(1H,m),7.32(2H,d,J=8.4Hz)

False positive: 1H-NMR(CDCl₃,TMS) δ (ppm):2.55(3H,s),6.75-6.85(1H,m),7

The errors produced by the missing separators in peak lists were simple to resolve, by making the presence of the separators in these lists optional. To produce regular expressions that thoroughly compensate for typos that authors may make would be inadvisable, however, since this would require the creation of regular expressions that can ignore much of the common structure that reports of spectra share – and that the strategy of regular expression matching exploits. Consequently, it would result in a far higher rate of false positives – increasing the recall of the system at the cost of precision.

Mass Spectrometry Peak Inversion

The format expected by OSCAR3 α 5 for reports of mass spectra is as follows;

m/z: 597 (M + H)⁺

That is to say, the fragment mass should precede the assignment (if present). A number of spectra found in this analysis, however, exhibited an inversion of this format, *i.e.* the assignment preceded the mass of the fragment, *e.g.* “MS: MH⁺ = 445.2” or “ESIMS (M+H)⁺ = 624”. This problem was solved by the creation of regular expressions to match this alternative template.

Mass Spectrometry Partial Annotation

It was found that frequently the OSCAR3 annotation would exclude part of the manual annotation.

For example;

Manual annotation: MS (ESI) m/z 467.5 [M+1]⁺

Automatic annotation: m/z 467.5 [M+1]⁺

Manual annotation: ESI-MSm/z: 455(M + H)⁺

Automatic annotation: m/z: 455(M + H)⁺

In these cases it was common that the part of the spectrum which to be omitted from the annotation, as in the examples above, was a description of the method by which the spectrum was measured. However, in such cases the peak(s) and assignment(s) were included in the annotation. For this reason, the correction of these errors was not attempted as the annotation obtained by OSCAR3, while technically incorrect according to the method of measurement chosen, was of sufficient accuracy for the practical purposes of this thesis.

Mass Spectrometry Assignments

When a mass spectrum is reported, it is common that the reported mass is not that of the molecule for which the spectrum is being recorded. As in the examples above, it is common to report instead the mass of the most intense peak in the spectrum and to assign to this mass a chemical species, *e.g.* “288[M+Na]” or “385 (M+1)”. In these cases, if the data identified in the source document is to be put to good use, it is necessary to capture this assignment in a machine-understandable way such that the reported mass becomes meaningful. This process was not supported in OSCAR3 in advance of the current work – consider the inline annotations produced by OSCAR3 α5 for the spectrum “MS m/z 362 (M+1)”;

```
<spectrum type="massSpec">
  MS m/z
  <peaks type="..">
    <peak>
      <quantity type="mass">
        <value>
          <point>362</point>
        </value>
      </quantity>
      (M+
      <quantity type="intensity">
        <value>
          <point>1</point>
        </value>
      </quantity>
    )
  </peak>
</peaks>
</spectrum>
```

While the assignment has been included in the spectrum annotation, it has not itself been specifically annotated. In advance of this work, the data regexes in OSCAR3 allowed virtually any bracketed text to follow the numerical value in a mass spec peak, and did not attempt to identify assignments in mass spectra. This has caused the numerical value, 1, in the assignment “M+1” to be interpreted as being the intensity of the given peak within the spectrum. Clearly, this is both mistaken and unhelpful for a client programmer who wishes to extract and work with the spectral data. To address this issue, an additional child of the `peak` node was created – the

massSpecAssignment. This required the creation of regular expressions to match specifically the forms of assignment that are used rather than accept generic text contained within brackets. For example, assignments of the form “[M-H]” are matched by the regular expression;

```
(?:
  [\\(\\{\\[\\
    M (?!\\s) [\\+<insert idref="HYPHENCHARACTERS"/>]*
    (?:
      (?x-i:<insert idref="ELEMENTS"/>\\d*)* | \\d+
    )
  [\\)\\}\\]]
  [\\+<insert idref="HYPHENCHARACTERS"/>]
)
(?:\\w)
```

As a result of this work, OSCAR3 now produces the following annotations for the previous example, “MS m/z 362 (M+1)”;

```
<spectrum type="massSpec">
  MS m/z
  <peaks type="..">
    <peak>
      <quantity type="mass">
        <value>
          <point>362</point>
        </value>
      </quantity>
      <quantity type="assignment">(M+1)</quantity>
    </peak>
  </peaks>
</spectrum>
```

With OSCAR3 now producing output in this format, it is possible for a machine to trivially identify that the peak at a mass of 362 has been given the assignment “M+1”, just as it was for a trained chemist to identify this information from the original text. This additional information can be used to predict the expected mass of the compound for which the spectrum was measured, as described in section 5.3.2.5.

5.2.3.4 Measurement of Improved OSCAR3 Performance

Once the work described in the previous section was completed, the analysis described previously was re-performed using the upgraded regular expressions. Results are also presented for the analysis of the validation corpus, using both the original and the updated regular expressions. For the purposes of comparison, the results of the analysis on the development corpus using the original regular expressions are reproduced;

Spectrum type	# in corpus	TP	FP	FN	Precision	Recall
MassSpec	227	104	51	123	67.1%	45.8%
HNMR	206	97	21	109	82.2%	47.1%
CNMR	12	6	2	6	75.0%	50.0%
IR	10	0	0	10		0.0%
HRMS	5	0	0	5		0.0%

Table 5-5: Performance of OSCAR3 α 5 on the development corpus

Spectrum type	# in corpus	TP	FP	FN	Precision	Recall
MassSpec	227	146	67	81	68.5%	64.3%
HNMR	206	175	16	31	91.6%	85.0%
CNMR	12	10	2	2	83.3%	83.3%
IR	10	0	0	10		0.0%
HRMS	5	0	3	5	0.0%	0.0%

Table 5-6: Performance of the improved OSCAR3 on the development corpus

Spectrum type	# in corpus	TP	FP	FN	Precision	Recall
MassSpec	199	80	55	119	59.3%	40.2%
HNMR	202	103	16	99	86.6%	51.0%
CNMR	24	12	2	12	85.7%	50.0%
IR	8	0	0	8		0.0%
HRMS	14	6	7	8	46.2%	42.9%

Table 5-7: Performance of OSCAR3 α 5 on the validation corpus

Spectrum type	# in corpus	TP	FP	FN	Precision	Recall
MassSpec	199	122	53	77	69.7%	61.3%
HNMR	202	179	40	23	81.7%	88.6%
CNMR	24	22	4	2	84.6%	91.7%
IR	8	0	0	8		0.0%
HRMS	14	6	7	8	46.2%	42.9%
UnknownNMR	-	-	3	-		

Table 5-8: Performance of the improved OSCAR3 on the validation corpus

The manually-annotated corpus does not, of course, contain any spectra of type ‘unknownNmr’. A strict application of the previous procedure therefore produces false positives wherever OSCAR3 annotates as such. The metrics produced in this way are misleading – if OSCAR3 annotates an NMR spectrum that fails to identify the isotope under investigation and if the correct text is annotated then it has correctly performed its function. As a result, the results given above have been computed allowing for a manually-annotated HNMR or CNMR spectrum to be matched by an automatically-annotated unknownNmr.

In the results given above, it can be seen that the performance of OSCAR3 on this task has been significantly improved by the work described previously. The recall on previously unseen data has been significantly improved (88.6% up from 51.0% for HNMR, 61.3% up from 40.2% for Mass Spec), while the precision has been only marginally affected (69.7% up from 59.3% for Mass Spec, 81.7% down from 86.6% for HNMR). These modifications have greatly increased the potential for using OSCAR3 as a means for large-scale automated collection of spectral data from the literature.

5.2.4 Experimental Paragraph Classification

While it is common for the experimental sections, *i.e.* those that describe the process and results of a chemical reaction, of a patent to occur as examples of the invention, it is not necessarily the case that the method of identifying document sections described in section 5.2.2.1 will result in their occurrence as part of an `example` element in the semantically enhanced patent documents. As a result, the semantic enhancement at this point has done nothing to identify the presence or location of some or all of the experimental sections in a number of documents. To address this concern the sections of the text, as contained by opening and closing `heading` tags, are classified as being either *experimental* or *non-experimental* by use of a naïve Bayesian classifier. This classification allows for a greater proportion of the experimental sections within the patent corpus to be recognised as such and treated appropriately during the later stages of the workflow.

5.2.4.1 Classifier Implementation

The implementation of the naïve Bayesian classifier used here is supplied by the `BayesianClassifier` class in the third-party Java library `Classifier4J` (101), version 0.6. The API provided by `BayesianClassifier` allows for the binary classification of text strings, *i.e.* belonging

or not belonging to a given class. This class is determined by the client programmer and is taught to the Bayesian classifier by the provision of a number of examples of matches and non-matches. Once the training process is complete, the classifier will predict the likelihood of a previously unseen strings belonging to the class.

5.2.4.2 Training and Validation

A corpus was assembled by selecting 800 p elements (*i.e.* paragraphs, in the most part) from those patents that had successfully passed through the paragraph deflating and document segmentation phases of the semantic enrichment procedure, using a random process in which each paragraph had an equal chance of selection. These paragraphs were given file names numbering them sequentially from para000 to para799, and were manually inspected and determined to be *experimental*, *non-experimental* or *empty* according to the following criteria;

- The paragraph is *empty* if it has no text content. Such empty paragraphs generally occur in the patent documents as containers for images.
- The paragraph is *experimental* if;
 - It is an account of a reaction or a part of a reaction, including by way of reference to another section of text *e.g.* "The reaction was carried out as in example 12".
 - It is a report of spectral or other characterisation data.
 - It is some combination of the above.
- The paragraph is *non-experimental* if it is not *empty* or *experimental*.

The manually-classified paragraphs may be summarised as follows;

Class	Frequency of Occurrence
Empty	117
Experimental	238
Non-experimental	445

In order to produce experimental and non-experimental sets of equal size, non-experimental paragraphs after the 238th were ignored for the remainder of this work. The first 119 (50% of the full set) experimental and non-experimental paragraphs were then used to train the Bayesian classifier before it was asked to predict probabilities of the remaining experimental and non-experimental paragraphs belonging to the experimental class. The predicted likelihoods may be summarised as follows;

Experimental	
Predicted likelihood	Frequency
0.99	115
$0.98 \geq p > 0.95$	1
$0.05 \geq p > 0$	4

Non-experimental	
Predicted likelihood	Frequency
0.01	102
$0.01 < p \leq 0.06$	3
$0.06 < p < 0.5$	2
0.99	12

Thus, when classifying paragraphs as experimental if $p < 0.5$ and non-experimental if $p > 0.5$, the experimental paragraphs were correctly classified at a rate of 96.6% and the non-experimental paragraphs at a rate of 89.9%. These rates were deemed high enough to continue into production.

5.2.4.3 Integration into Workflow

The naïve Bayesian classifier is integrated into the PatentEye code through the `ParagraphClassifier` class. This class handles the loading from disk of the training data described in the previous section as well as the training of the `BayesianClassifier` and delegating calls to classify sections of text to the underlying implementation.

`Heading` elements in the patent documents are identified by use of the XPath `“//heading”`. If a heading has text content, the text content is passed to the `ParagraphClassifier` for a prediction to be made. If the predicted likelihood is greater than 0.5, the section is classified as being experimental, and this is noted in the XML by the addition of a `classifier4j` attribute with the value `“experimental”`. Otherwise, the opposite is recorded by setting the value of the `classifier4j` attribute to `“nonExperimental”`.

5.2.5 Image Analysis

As previously discussed, the XML documents supplied by the EPO frequently employ images to communicate chemical information, including Markush structures, reaction schemes and illustrations of single chemical structures. These images contain important information in the context of understanding the content of the document, but this information is unavailable to a machine. It is desirable for PatentEye to identify the compounds represented in structural diagrams where possible so that they may be used to confirm or to query the identity of the example compounds of the patent. The images are supplied in TIF format, and are embedded in the XML document as follows;

```

<p id="p0473" num="0473">
  <chemistry id="chem0413" num="0413">
    <img id="ib0413" file="imgb0413.tif" wi="87" he="48"
      img-content="chem" img-format="tif" orientation="portrait"
      inline="no" />
  </chemistry>
  Diisobutylaluminium hydride (7mL, 1 M in tetrahydrofuran , 7mmol)
  was added to a cooled (-10°C) solution of the ester from preparation
  184 (1.2g, 2.7mmol) in tetrahydrofuran (25mL), and the reaction
  stirred for an hour at -10°C, followed by 1Hour at 0°C. Tlc analysis
  showed starting material remaining, so additional
  diisobutylaluminium hydride (5.4mL, 1 M in tetrahydrofuran, 5.4mmol)
  was added and the reaction stirred at 10°C for 10 minutes.
</p>

```

Figure 5-5: Embedded images in the patent XML. The text has been shortened for the sake of brevity

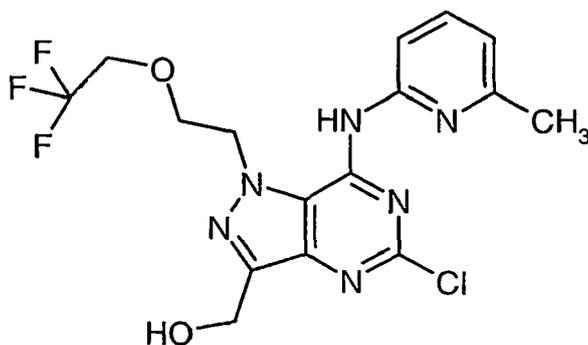


Figure 5-6: EP1620437B1 Image 413

In this case, the image is a chemical structure diagram for the product of the reaction. The information contained within such images may be crucial in correctly identifying the product of a reaction. Mention is made in the DTD for the EPO patents that `chemistry` elements may at some point in the future support the inclusion of CML, but for the moment to facilitate the usage of this information it is necessary to first convert them into a machine-understandable format. As previously discussed, this problem has been previously addressed and applications for this purpose have been developed. The generation of connection tables for the images embedded within the

patents is achieved by interfacing with the application OSRA (65; 66; 67), and this process is subsequently discussed.

5.2.5.1 Technical Aspects

OSRA version 1.2.2 was used for the current work, which was the most recent version at the time that the work commenced. OSRA is implemented in C++ and is distributed as a pre-compiled executable for the Windows environment. It operates as a command line facility, and using default parameters the command "osra <filename>" instructs the program to process the specified files and print at the command line a series of line-separated SMILES strings for the chemical structures found within it. The output produced by OSRA is then added to the patent XML as an attribute on the `img` element. For example, the enhancement produces, for the example from Figure 5-5, the following output;

```
<p id="p0473" num="0473">
  <chemistry id="chem0413" num="0413">
    <img id="ib0413" file="imgb0413.tif" wi="87" he="48"
      img-content="chem" img-format="tif" orientation="portrait"
      inline="no"
      osraResult=" OCc1nn (CCOCC (F) (F) F) c2c (Nc3cccc (C) n3) nc (Cl) nc12"/>
    </chemistry>
    Diisobutylaluminium hydride (7mL, 1 M in tetrahydrofuran , 7mmol)
    was added to a cooled (-10°C) solution of the ester from preparation
    184 (1.2g, 2.7mmol) in tetrahydrofuran (25mL), and the reaction
    stirred for an hour at -10°C, followed by 1Hour at 0°C. Tlc analysis
    showed starting material remaining, so additional
    diisobutylaluminium hydride (5.4mL, 1 M in tetrahydrofuran, 5.4mmol)
    was added and the reaction stirred at 10°C for 10 minutes.
  </p>
```

Where OSRA identified the presence of more than one structure in an image, it returns a set of line-separated SMILES strings. So that they may be embedded into the XML document, this set of SMILES strings are concatenated into a single pipe ("|") separated string that can be used as the value for a single `osraResult` attribute. While it is possible to produce a single, valid SMILES string that represents two or more disconnected molecular structures by using the dot character ("."), e.g.

“CC(=O)O.CCO” to represent acetic acid and ethanol, the dot-separated molecular graphs can be reconnected by using ring closures, *e.g.* “C1CCCCC1” represents cyclohexane and “C1CC.CC1” represents hexane. In order to prevent the possibility of malformed SMILES output from OSRA effecting bond formation between disconnected structures, it was decided to use the pipe character, which does not occur in the SMILES vocabulary, to denote separated structures.

Since OSRA is not implemented in Java, the PatentEye application interacts with it by using the capacity provided by the Java system libraries to execute arbitrary applications, wait for their termination and read into memory any output produced. The generation of connection tables from image files using OSRA is a time-consuming operation, as discussed in section 5.2.5.3, and so the results that are generated by OSRA are immediately cached and written to disk such that the task need not be repeated. In order to reduce the amount of time taken to parse the images for a given patent, only a subset of the images present in a patent undergoes the image analysis procedure. This subset is composed of those images that are selected by the XPathS “//example//chemistry/img” and “//heading[@classifier4j='experimental']//chemistry/img”, *i.e.* those that are embedded in `img` tags that are children of a `chemistry` element that in turn is a descendant of an `example` element or a `heading`, the content of which has been classified as experimental. This selection is performed for a practical reason – only the images in these positions will be used later to corroborate a candidate product for a reaction, and so to analyse other images would be to waste computing resources.

OSRA is claimed to support the TIF format, but it was discovered that an apparent bug in OSRA version 1.2.2 prevents the direct interpretation of TIF files. To work around this, the application ImageMagick (102) is first used to convert the TIF images into the PNG format, which OSRA accepts as input. ImageMagick is executed using the same procedure for calling external applications as described previously for OSRA.

5.2.5.2 OSRA Performance

It was desired to validate the performance of OSRA in the context of this work. To produce a corpus reflective of the task in question the complete set of images contained within experimental sections of the ten-week corpus of semantically enhanced patents were identified using the XPaths given above, and from these 14697 images a subset of 300 were randomly selected. The TIF files for these images were extracted from their parent ZIP files, numbered sequentially and manually inspected. These images were found to conform to one of a number of different types, by far the most common of which was of a single chemical structure diagram. The set of 300 images was used to create a corpus of 200 images of single chemical structures by selecting, in ascending order, the first 200 such images. The types of images contained in the range image0 to image271 – image271 being the 200th single chemical structure image – are summarised in Figure 5-7.

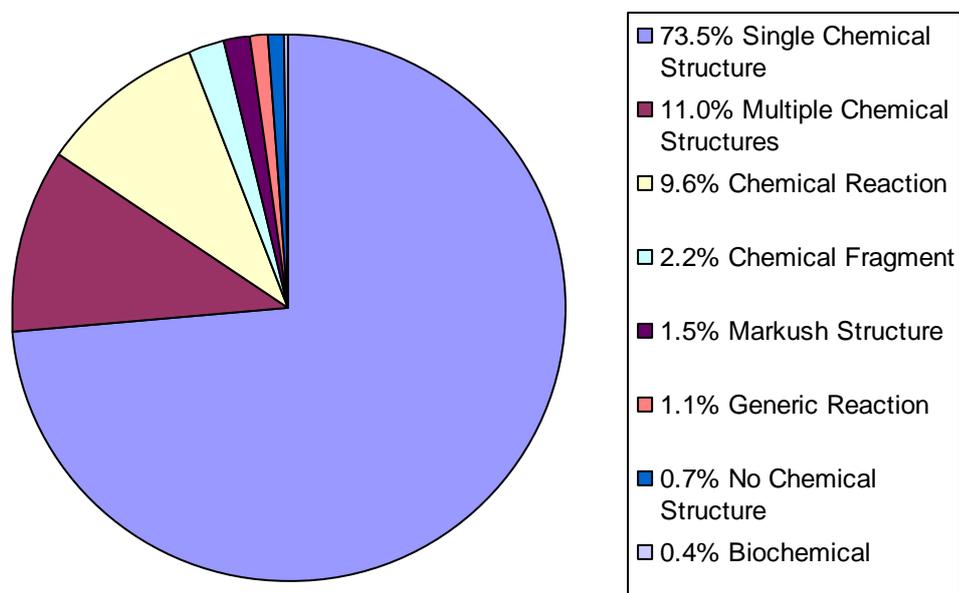


Figure 5-7: Types of images present in experimental sections

The chemical structures contained within the set of 200 single chemical structure images were manually converted to SMILES strings, chiefly by redrawing the structure with ChemDraw 12.0 (35) and exporting the structure as SMILES or by manual conversion in the case of simple structures, which were recorded in an index of the corpus. OSRA was used to analyse each of the 200 single chemical structure images, and the results of this analysis was appended to the index.

Previous authors in the field have suggested subjective metrics of success such as less than 30 seconds of human editing being required to correct errors in the structure (61), while Filippov and Nicklaus propose measuring success by calculating a similarity metric between the machine-produced structure and the correct structure. Such measures are of limited utility in the present work; manual correction of structures or determination of correct structures cannot be implemented within a fully automated workflow. What is desired of the image analysis process is the correct identification of the product molecules of chemical syntheses, and while a high similarity between a structure believed to be the product (the “candidate product”) and a structure produced by OSRA may be indicative that the image analysis has made a minor error and the candidate product should be accepted, it may equally indicate that the image analysis is correct and the candidate product should be rejected. As a result, there is no threshold of similarity below the two structures being identical at which the structure derived from the image analysis becomes “good enough”.

The manually-generated and OSRA-generated SMILES strings for each image were thus used to generate InChIs using JUMBO. The performance of OSRA was measured by comparing these InChIs by string equivalence; where the two InChIs were identical, it was counted as OSRA having correctly deduced the chemical structure contained within the image and considered a match. Where the InChIs differed it was considered a non-match. As discussed in section 2.2.3.2, InChIs are composed of layers that describe the molecule in increasing levels of detail, and so the two were examined

side-by-side to determine the level of agreement, *i.e.* the highest layer of the InChIs at which the two disagreed. In a number of cases, it was not possible to generate an InChI from the SMILES string produced by OSRA. The causes of these problems were also examined and determined to be primarily that the SMILES string contained the wildcard character, *, which is valid SMILES but is not supported by JUMBO or by InChI. In a further two cases the SMILES string returned by OSRA was found not to be valid, suggesting a bug within the OSRA program itself.

The results from this work were as follows;

Result	Frequency	%
Match	68	34.0
Non-match	79	39.5
Unbuildable SMILES (containing wildcard)	51	25.5
Invalid SMILES	2	1.0

Table 5-9: OSRA performance

Cause	Frequency	%
Differing Hydrogen Count	12	15.2
Differing Molecular Formula (excluding H count)	57	72.2
Differing Regioisomers	4	5.1
Differing Stereochemistry	5	6.3
Differing Charges	1	1.3

Table 5-10: Causes of InChI disagreement

The rate at which the OSRA-produced structure and the manually-produced structure is, at 34%, significantly lower than that reported for OSRA 1.1.0 by Filippov and Nicklaus (103), in which the rate was reported as 26 matches out of 42 (61.9%) structures and 107 matches out of 215 (50.0%) structures on two data sets. Such rates will of course be highly dependent upon the images that form the test corpus, and the images supplied by the EPO are of highly variable quality. Many of the images that form the test corpus used in this work are severely pixelated, indistinct or contain background noise; some are only barely legible to a human skilled in the art. Examples from the single chemical image corpus are subsequently illustrated, along with the resultant structures produced by OSRA. All input images are the work of the European Patent Office and the original patent authors.

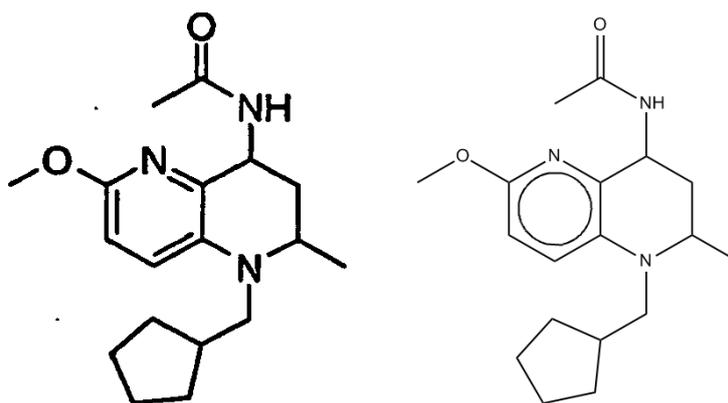


Figure 5-8: Input image (left) and correctly interpreted structure (right)

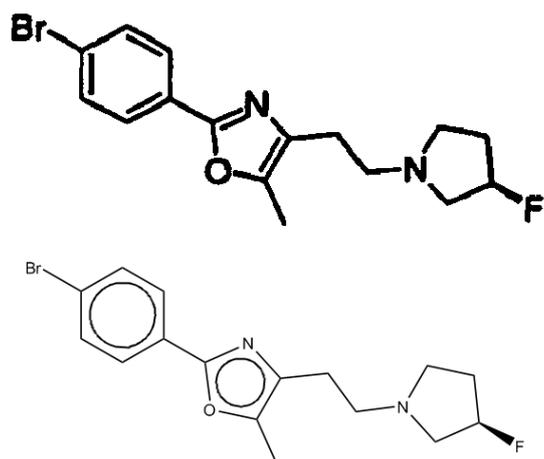


Figure 5-9: Input image (top) and correctly interpreted structure (bottom)

Figure 5-8 and Figure 5-9 show examples of images from the test corpus that were correctly interpreted by OSRA. Note that in Figure 5-9 the wedge bond to fluorine was correctly interpreted even though it is not immediately noticeable to the human eye.

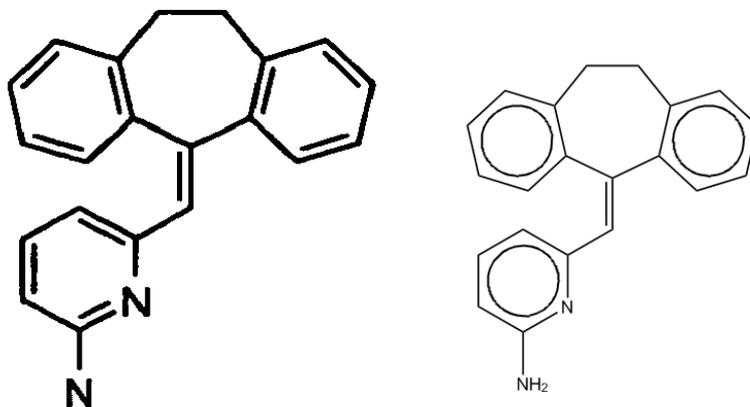


Figure 5-10: Input image (left) and correctly interpreted structure

In Figure 5-10 we see that although the image contains a mistake – the missing hydrogen atoms in the amine substituent – OSRA has correctly identified the structure in the image.

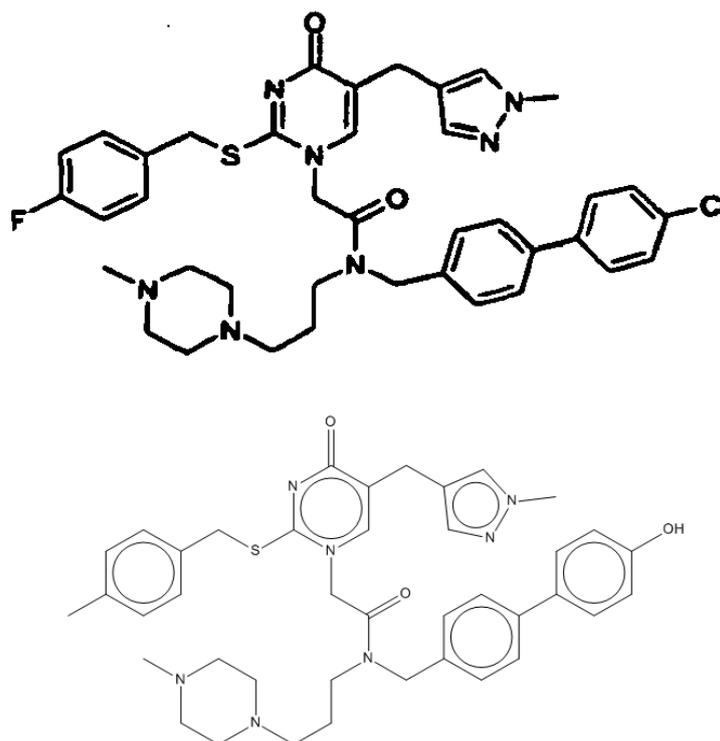


Figure 5-11: Input image (top) and incorrectly interpreted structure (bottom)

Figure 5-11 shows an example of OSRA producing buildable but incorrect output. The label for the fluorine atom has been missed, giving a methyl substituent on the benzene ring instead of the correct fluorine substituent. Furthermore, the chlorine substituent has been misrecognised as a hydroxyl group, presumably due to the letters “C” and “l” overlapping slightly to be recognised as an “O”, and the hydrogen being added automatically as in Figure 5-10. The analysis of the causes of structural disagreements such as these, presented in Table 5-10, suggest that when these errors occur they are major errors, with 15.2% being due to discrepancies in the hydrogen count of the molecules, and a further 72.2% being due to other discrepancies in the molecular formula, such as in Figure 5-11.

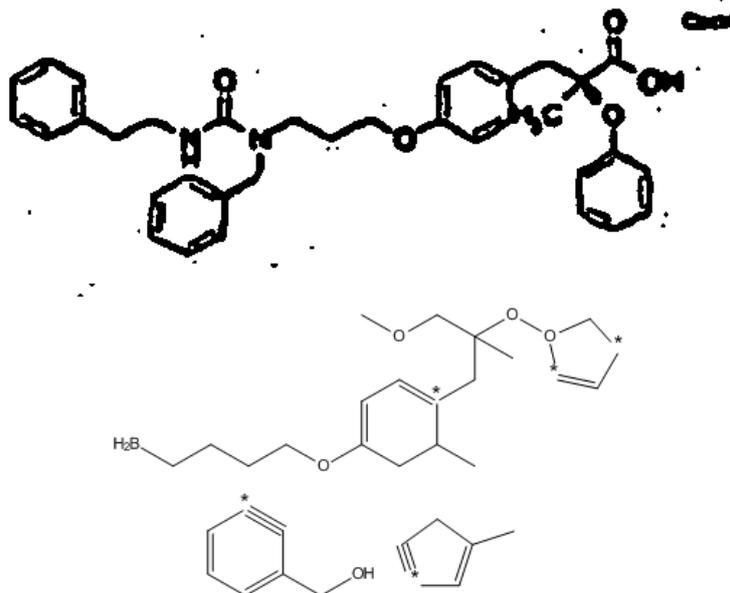


Figure 5-12: Input image (top) and unbuildable result (bottom)

In Figure 5-12, the input image is of extremely low quality. Consequently, the output produced by OSRA comprises three disconnected subsections of the molecule and it is difficult to identify which sections of the output structures correspond to which sections of the input structure. The presence of the wildcard character, “*”, in the structure generated by the image analysis is generally indicative that OSRA has not correctly identified the full chemical structure and is returning a partial result. Consequently, it is evident that the result returned by OSRA is not a correct representation of the structure contained in the image and may be safely disregarded in any workflow.

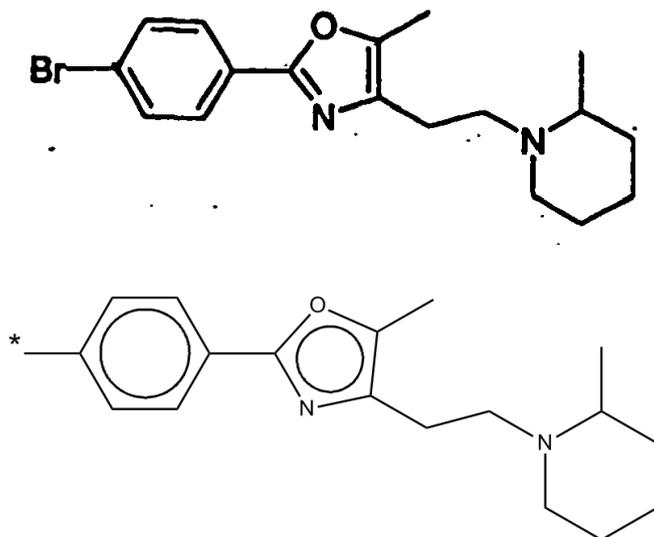


Figure 5-13: Input image (top) and unbuildable result (bottom)

Figure 5-13 shows a further example in which OSRA returns a partial interpretation of the input image. In this case, the structure produced by OSRA is entirely correct apart aside from the missing bromine atom. In such cases it is possible to determine that a candidate product subsumes the partial structure defined by the OSRA result, and thus that the structure determined by the image analysis agrees with the candidate structure *to the fullest extent possible*. It was not, however, attempted to implement this process within the current work.

5.2.5.3 Computational Expense

The analysis of the illustrative images was the most computationally expensive part of the PatentEye workflow, but was still capable of running on a standard desktop PC (Pentium 4, 3.40 GHz). The time taken to compute a SMILES string from an input image on this platform varied from 890 milliseconds to 12,026,543 milliseconds – over three hours. The variation in time required for this task is illustrated in Figure 5-14.

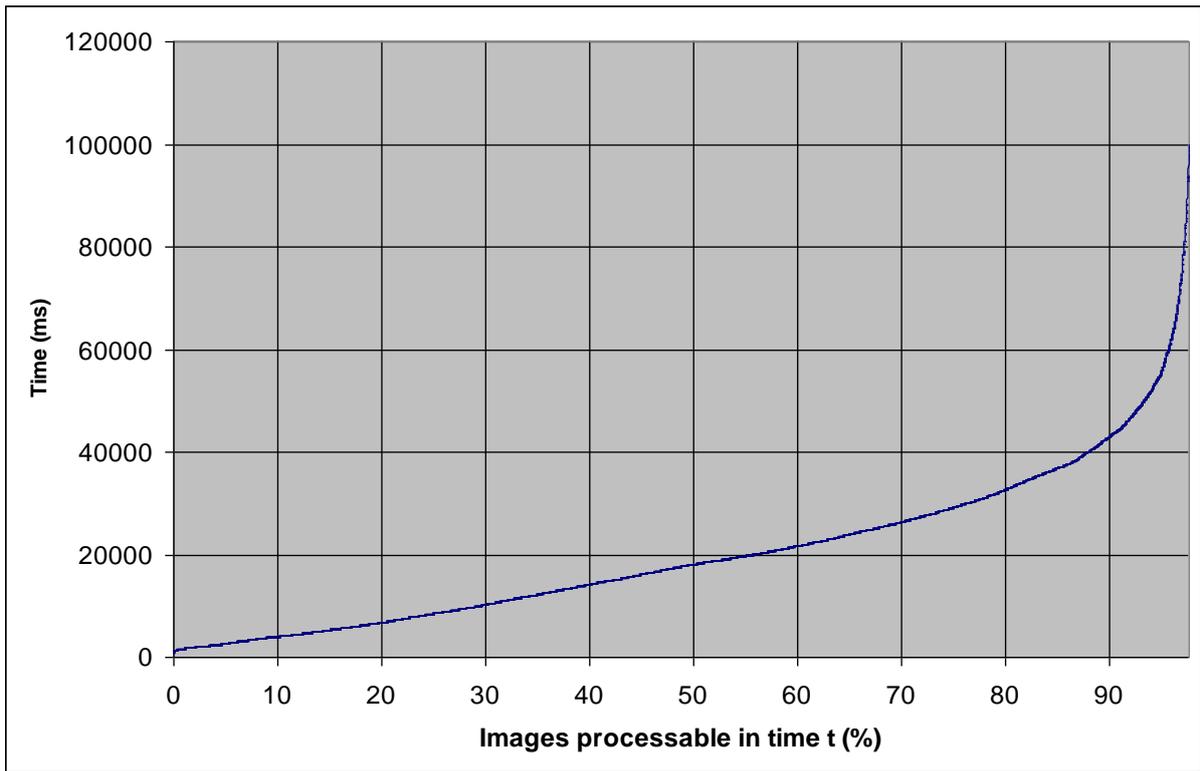


Figure 5-14: Runtime required for image analysis

To produce a legible graph, Figure 5-14 ignores the 216 images for which the required runtime exceeded 100 seconds. The graph of the remaining 9271 (97.7% of the total) images that were processable in under 100 seconds show a roughly linear increase in required runtime over the fastest 80-90% before the required runtime begins to increase markedly. The fastest 80% of the images may be processed in less than around 32 seconds each, while the fastest 90% may be processed in less than around 42 seconds each. This shows that the existing software tools and standard hardware may compute the bulk of the information that is to be gained from the images within reasonable periods of time.

5.2.6 Back Reference Annotation

By the nature of a chemical patent document, the examples of the invention frequently consist of the syntheses of a number of related compounds. The methods of synthesis of these compounds are frequently the same, producing differing products by varying one or more of the reactants.

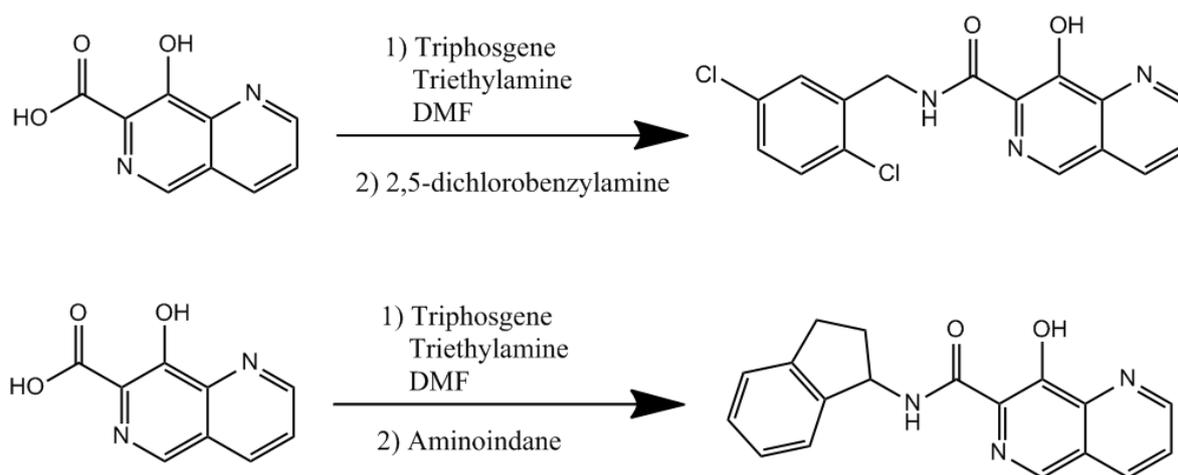


Figure 5-15: Analogous reactions from EP1326865

This phenomenon is illustrated in Figure 5-15. The second reaction is described in the patent document as follows;

“The title compound was prepared using the procedure described in Example 2, Step 2 replacing 2,5 dichlorobenzylamine with 1(R,S) aminoindane.”

The patent author has saved himself from the need to repeat the description of procedure, conditions and reagents that has already been included for the first reaction. Authors also frequently refer to a reactant as being “the ester from step 3”, or similar, instead of by name. For a machine reading the document, these constructions create a problem – the information necessary to understand a reaction is not present in the description of that reaction, instead being present elsewhere in the document. A human reader will recognise the intent of the *Back Reference* to the earlier section of the document and will understand that he should refer back

to obtain the information that has been omitted from the immediate section – indeed, this is what happened during the process of the drawing of Figure 5-15 – but for a machine this process is not so trivial.

In order to facilitate the same process to be performed automatically, the document is searched for text strings that may be such references to previous sections. These *Back References*, where found, are annotated in the semantically enhanced documents with a link back to the referenced section so that the machine will later be able to look up the relevant information when attempting to process the reactions described in the patent. The process of *Back Reference* annotation is subsequently discussed.

5.2.6.1 Hierarchical Indexing of Reaction Headings

The document to be annotated is passed to the `BackReferenceAnnotator` class, which is intended to identify and annotate potential back references in the input document. Before any annotation can be carried out, the `BackReferenceAnnotator` must first identify the terms to be annotated. This is achieved by locating the `example` headings in the document using XPath and tokenising the title text of the headings on whitespace. Subheadings, *i.e.* `heading` or `example` elements that are children of the previously indexed `example` element, are similarly indexed and are recorded in the index of headings as children of their parent heading. For example, the following document structure would be indexed as shown;

```

<example id="h0005" title="EXAMPLE 1">
  <heading id="h0006"
    title="N-(3,5-dichlorobenzyl)-8-hydroxy-1,6-naphthyridine-7-
    carboxamide">
    <p id="p0135" num="0135">...</p>
  </heading>
  <heading id="h0007"
    title="Step 1: Preparation of 3-{[Methoxycarbonylmethyl-(toluene-
    4-sulfonyl)-amino]-methyl}-pyridine-2-carboxylic acid isopropyl
    ester">
    <p id="p0136" num="0136">...</p>
  </heading>
  <heading id="h0008"
    title="Step 2: Preparation of methyl 8-hydroxy-1,6-naphthyridine-
    7-carboxylate">
    <p id="p0137" num="0137">...</p>
  </heading>
  <heading id="h0009"
    title="Step 3: Preparation of N-(3,5-dichlorobenzyl)-8-hydroxy-
    1,6-naphthyridine-7-carboxamide">
    <p id="p0138" num="0138">...</p>
  </heading>
</example>

```

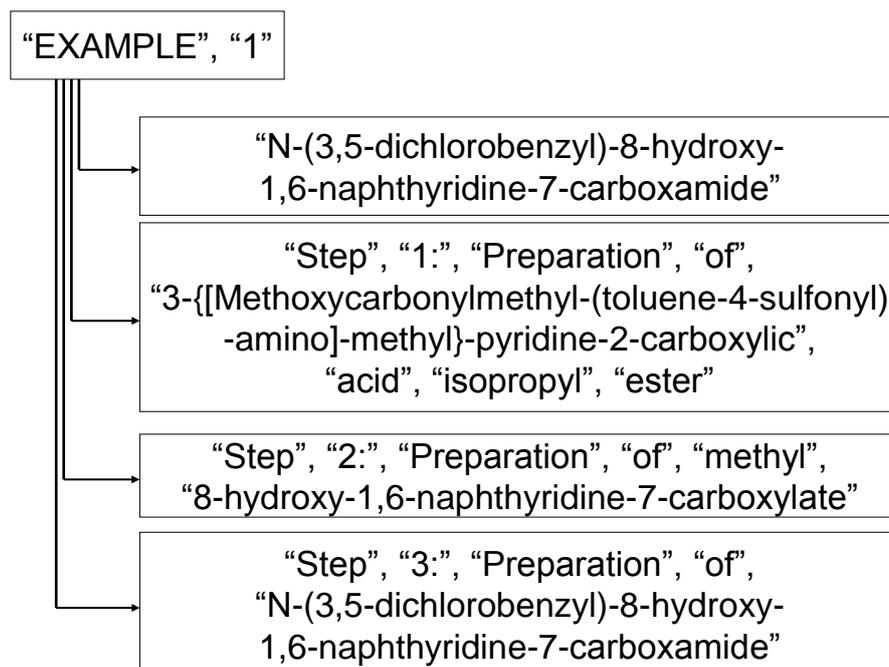


Figure 5-16: Indexed and Tokenised Headings

Figure 5-16 shows the index generated from the preceding XML. Note that the structure of the XML has been preserved in the index, with one example heading containing four child headings. For each of the indexed headings, the tokens generated by tokenising the title text on whitespace are shown

within quotation marks. The index generated in this way is subsequently used to annotate references to these sections of text in the patent text.

5.2.6.2 Text Annotation

Paragraphs that are contained by example headings or by headings that have been classified as experimental by the `ParagraphClassifier` are identified within the patent document using XPath. The text content of these paragraphs is tokenised on whitespace, as was the text of the headings, and then the token sets are compared to identify common tokens between the two. Common tokens in this case are identified by allowing for differences in case and for the punctuation at the end of the tokens to differ, such that the heading for “EXAMPLE 1” in the previous example would find a match within the string “Example 1, Step 2”. To avoid the annotation of common strings such as “1” or “the” that do not indicate the presence of a *Back Reference*, matches must be of two or more consecutive tokens from the heading.

In the previous example paragraph text, “Example 1, Step 2”, it is clear that the reference is not to the whole of example 1, but specifically to the second step of that example. This is identified by the `BackReferenceAnnotator` by allowing the child headings of a parent heading to attempt to continue matching from the position in the paragraph token set where parent heading stopped matching.

Once this process of identifying the paragraph text that matches the indexed headings is complete, the match is checked to see if it is a *Compound Reference* – a specific type of *Back Reference* that indicates the use of a previously defined chemical *e.g.* “the product of example 3”. This is achieved by checking if the text that precedes the matched text itself matches the regular expression;

```
the (.*) (from|of|(synthesi[sz]ed|produced) in)$
```

Furthermore, the text matched by the capture group (.*) can either be the word “product” or a single entity of type CM as annotated by OSCAR3. The preceding text is therefore required to be of the form “the aldehyde from” or “the product synthesised in”, with the anchor character “\$” requiring that the matching text occurs at the end of the string. If the preceding text matches this regex then both matches are annotated as a single *Compound Reference*, otherwise the text matched to the example headings is annotated as a *Back Reference*.

Compound References are sometimes used by patent authors with a local rather than global scope – that is to say, the compound referred to by a certain phrase will not be the same as the compound referred to by the same phrase in another part of the document. This is observed, for example, in the phrase “the product of step 1” – it is assumed and inferred that this refers to the first step of the current reaction. For this reason, the *BackReferenceAnnotator* also annotates *Compound References* by identifying sections of text that match to headings that are siblings of the parent heading of the paragraph containing the text. This is illustrated in Figure 5-17, where the colour coding indicates the section of the document for which each paragraph is annotated.

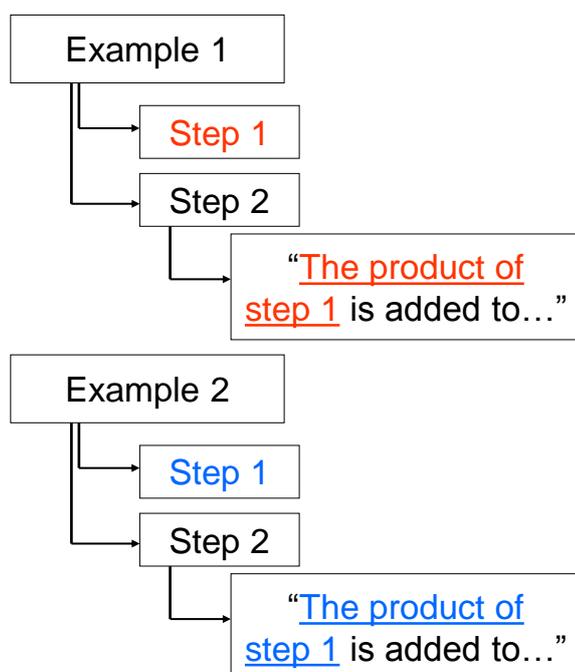


Figure 5-17: Local annotation of sub-headings

The XML produced by this process is of the following form;

```
<p id="p0141" num="0141">
  Triphosgene (0.556g, 1.87 mmol) was added over 20 mins to a
  solution of <compoundRef targetId="h0012">the acid from step
  1.</compoundRef> (0.89g, 4.68 mmol) and diisopropylethylamine 3.26
  ml, 18.7 mmol) in DMF (22 ml) at 0°C.
</p>

<p id="p0143" num="0143">
  The title compound was prepared using the procedure described in
  <backReference targetId="h0013">Example 2, Step 2</backReference>
  replacing 2,5 dichlorobenzylamine with 1(R,S) aminoindane.
</p>
```

In each of these cases, the value of the `targetId` attribute on the reference annotation is equal to the value of the `id` attribute on the heading element to which the reference points, introducing into the document a means by which a machine may easily resolve the reference.

5.3 Extraction of Reactions

Chemical patents are a rich source of chemical reactions due to the requirement for a patent claimant to detail examples of the invention. The reactions published in this way are routinely manually indexed and added to databases such as CASREACT (4). A project at the Chemical Abstracts Service (CAS) in the 1980's aimed to produce a system capable of automating or partially automating the indexing process by application of Natural Language Processing (NLP) technologies (104; 105; 106). These works claimed to "satisfactorily" process 36 out of 40 synthetic paragraphs from the Journal of Organic Chemistry (104) and to produce "usable results" for 80-90% for simple synthesis paragraphs and 60-70% for complex paragraphs (106), where complex paragraphs are defined as describing general procedures, instances of general procedures, analogous syntheses and parallel syntheses. The size of the corpus used to produce this second set of results was not given, nor in

either case was the procedure used for corpus creation. Accordingly, it is not possible to regard this area as a solved problem.

The work at CAS produced a system capable of summarising a reaction by identifying its product, yield and component reaction steps. Discrete steps in the reaction were identified by the occurrence of verbs in the text, with each verb indicating an event and mapped to one of eight types of events; COMBINE, REACT, PREPARE, WORKUP, RESULT, TITLE, MISC and UNKNOWN. Within these events, further information such as chemicals, their amounts and roles, and reaction conditions were identified. Chemical syntheses were thereby extracted from the text as a sequence of the identified events.

The era in which this technology was developed was very different. As a division of the American Chemical Society, CAS was in the privileged position of having access to a large body of published work in an electronic format. The situation today is different – the ubiquity of electronic publication and explosion of the scale of publication has granted such access far more widely, though publishers may very well supply the works subject to restrictive terms of use. The EPO patents, however, are subject to no such restrictions and so the time for a re-examination of the subject of automated extraction of chemical reactions has come.

5.3.1 Conventional Format of Experimental Sections

In order to devise a system for automated extraction from reported syntheses, it is important to first consider the nature and common structure of such text. Fortunately, the reporting of chemical syntheses is highly stylised. By convention, chemists report syntheses using the past tense and the agentless passive voice. Consider the following;

Step 1: Preparation of 5-iodo-8-(1-phenyl-methanoyloxy)-[1,6]naphthyridine-7-carboxylic acid methyl ester

To a solution of 8-hydroxy-5-iodo-[1,6]naphthyridine-7-carboxylic acid methyl ester (9.41 g, 28.5 mmol, from Example 112 Step 1) and cesium carbonate (13.9 g, 42.8 mmol) in dry DMF (250 ml), was added benzoic anhydride (9.67g, 42.8 mmol) and the solution stirred at room temperature for 4 hours. The solvent was removed under reduced pressure and the residue was partitioned between saturated aq. ammonium chloride and extracted into CHCl₃. The combined organic extracts were washed with brine, dried over Na₂SO₄, filtered and the solvent was evaporated in vacuo. The residue was purified by flash chromatography (40% EtOAc/Hexane gradient elution switching to 1% MeOH/CHCl₃) to provide the desired product as a yellow solid.

¹H NMR (CDCl₃, 400MHz) δ 9.11 (1H, d, J=4.21Hz), 8.48 (1H, d, J=8.51 Hz), 8.32 (2H, d, J=8.33 Hz), 7.75-7.67 (2H,m), 7.56 (2H, t, J=7.69 Hz), and 3.93 (3H, s) ppm.

Figure 5-18: EP1326865 - Example 79, Step 1

Such descriptions of syntheses may be conceptually divided into three parts – the *primary reaction*, in which the target compound is completely or substantially produced; the *work-up*, in which the reaction is quenched and neutralised, solvents are removed, the product purified and suchlike; and the *characterisation*, in which spectral data is afforded to demonstrate that the product of the reaction is that intended. In the description of the primary reaction, reactants (“a substance that is consumed during the course of a reaction” (107)) are detailed by giving a name or other reference (e.g. “ketone 12b” or “the compound from step 2”) together with the quantity used, generally state by mass and by molar amount. Solvents are typically detailed by giving a name and the volume used. In the description of the work-up these quantities are commonly omitted, as in Figure 5-18. The identity of the product of the synthesis may be specified in one of two typical ways; in the heading of the section, as in Figure 5-18, or by statement at the end of the description of the work-up, e.g. “to yield 1,6-naphthyridine-8-carboxylic acid”.

5.3.2 Implementation of Automatic Reaction Extraction

A system for the automated extraction of reaction information from the EPO patents is implemented. While this system is implemented in such a way as to integrate directly with the

enhanced patents produced by the process discussed in section 5.2, the methods and software employed are generic and could be reused to produce a system suitable for alternative inputs. The operation of the system is summarised in Figure 5-19.

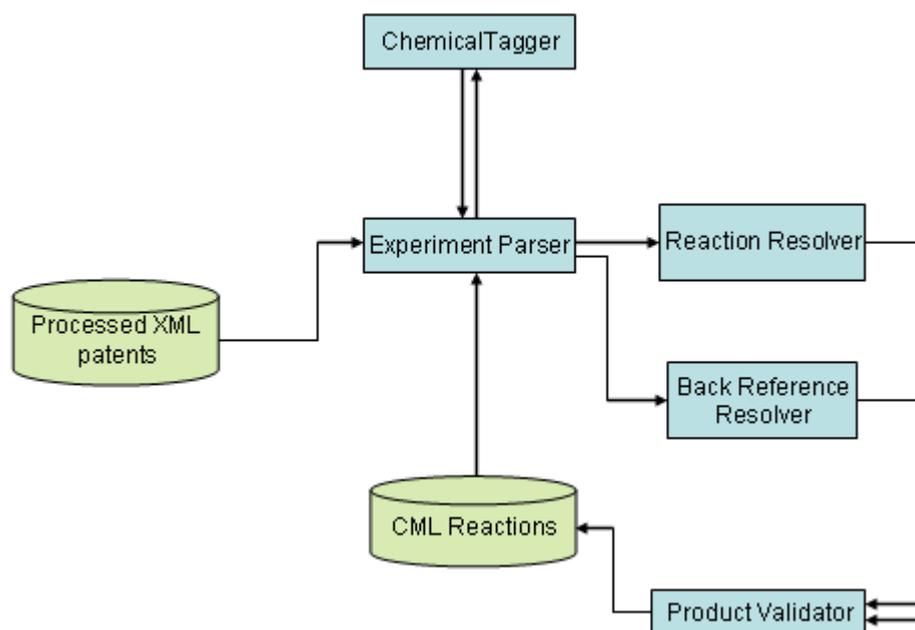


Figure 5-19: Abstracting reactions from patent text

The enhanced patent XML documents are read into memory, and the headings that have been classed as experimental by the ParagraphClassifier as well as those converted into `example` elements are identified by means of XPath. The sections of the document either contained within the `heading` or `example` element or, if the heading has sub-headings, each subheading individually, are passed into the ExperimentParser class. Identities and amounts of reagents are identified either by passing the text to ChemicalTagger, or by analogy with a previous reaction if a back reference is present. The DataAnnotator class in OSCAR3 is used to identify spectral data, and NMR spectra are converted to CML using a converter from JUMBOConverters (45). The product of the reaction is identified by using OSCAR3 to identify chemical names in the heading title. The product identity is then validated by comparison with the results of the OSRA analysis of any image present

(see section 5.2.5), and with any ^1H NMR or mass spectrum that is reported. The results of these processes are combined into a CML Reaction which is saved to disk and retained in memory in order that further reactions which refer back to this reaction may be resolved. This process is subsequently discussed in greater depth.

5.3.2.1 Identification of Products

In order to identify the product of a reaction, the title text of the document section under examination is passed to OSCAR3 for named entity annotation. If OSCAR3 does not identify a single named entity of type CM in the title text, then the process of reaction extraction fails and the ExperimentParser throws a RuntimeException. If a single CM is found in the title text, then the name is resolved to a CML Molecule, which is added to the productList of the CML Reaction.

5.3.2.2 Attachment of Spectral Data

As discussed in section 5.2.3, the most common spectra types found in the patent corpus were ^1H NMR, ^{13}C NMR and mass spectra. The reports of mass spectra generally report only the mass of the molecular ion, optionally plus or minus a defined offset, and so provide a useful source of information for validating a candidate product molecule but little information worth preserving – it is, after all, a simple task to calculate a molecular mass and the patents do not report fragmentation patterns. The NMR spectra, however, in addition to providing a means by which the product molecule may be verified are themselves data of potential importance and are worth preserving for future re-use. The format in which they are preserved in the enhanced patent XML documents, using inline annotation to identify features within the original patent text, is ideal in that context as it retains the original document text. It does not, however, enable trivial machine interpretation of the spectrum since it is not valid CML and tools do not exist for its easy manipulation. The OSCAR

annotated spectrum is therefore converted into a CML Spectrum by use of the `OSCAR2CMLSpectConverter` class in the `JUMBOConverters` library. This converter was created specifically for this task and was largely written by Peter Murray-Rust with some contribution from the present author. It does not attempt to perform any further text-mining on its input, instead relying entirely on the OSCAR3 annotations to fully identify features of interest such as peaks, integrals, multiplicities and coupling constants. Example output from this converter is given in Figure 5-20.

```

<spectrum xmlns="http://www.xml-cml.org/schema" type="nmr">
  <parameterList>
    <parameter dictRef="cmlx:solvent">
      <molecule>
        <name>CDCl3</name>
      </molecule>
    </parameter>
    <parameter dictRef="cmlx:frequency">
      <scalar dataType="xsd:double" units="unit:mhz">
        400.0
      </scalar>
    </parameter>
  </parameterList>
  <peakList>
    <peak xValue="9.18" integral="1.0" yUnits="unit:hydrogen"
      peakMultiplicity="cmlx:doubletdoublet">
      <peakStructure type="coupling" value="1.6" units="unit:hz" />
      <peakStructure type="coupling" value="4.2" units="unit:hz" />
    </peak>
    <peak xValue="8.53" integral="1.0" yUnits="unit:hydrogen"
      peakMultiplicity="cmlx:doubletdoublet">
      <peakStructure type="coupling" value="1.6" units="unit:hz" />
      <peakStructure type="coupling" value="8.5" units="unit:hz" />
    </peak>
    <peak xValue="8.26" integral="1.0" yUnits="unit:hydrogen"
      peakMultiplicity="cmlx:multiplet" />
    <peak xValue="7.72" integral="1.0" yUnits="unit:hydrogen"
      peakMultiplicity="cmlx:doubletdoublet">
      <peakStructure type="coupling" value="4.2" units="unit:hz" />
      <peakStructure type="coupling" value="8.5" units="unit:hz" />
    </peak>
    <peak xMin="6.84" xMax="7.04" integral="3.0"
      yUnits="unit:hydrogen" peakMultiplicity="cmlx:multiplet" />
    <peak xValue="4.72" integral="2.0" yUnits="unit:hydrogen"
      peakMultiplicity="cmlx:doublet">
      <peakStructure type="coupling" value="6.2" units="unit:hz" />
    </peak>
    <peak xValue="3.97" integral="3.0" yUnits="unit:hydrogen"
      peakMultiplicity="cmlx:singlet" />
    <peak xValue="3.89" integral="3.0" yUnits="unit:hydrogen"
      peakMultiplicity="cmlx:singlet" />
  </peakList>
</spectrum>

```

Figure 5-20: Example NMR spectrum in CML

5.3.2.3 Identification of Reagents

As previously discussed, reagents used during the primary reaction section of a chemical synthesis are, by convention, reported along with the quantity used. Such lexical patterns are easily identified using ChemicalTagger – the output format is as follows;

```

<MOLECULE>
  <OSCAR-CM>4-(dimethylamino)-benzenethiol</OSCAR-CM>
  <QUANTITY>
    <_LRB-> (</_LRB->
    <MASS>
      <CD>.147</CD>
      <NN-MASS>g</NN-MASS>
    </MASS>
    <COMMA>,</COMMA>
    <AMOUNT>
      <CD>.96</CD>
      <NN-AMOUNT>mmol</NN-AMOUNT>
    </AMOUNT>
    <_RRB->)</_RRB->
  </QUANTITY>
</MOLECULE>

```

Figure 5-21: Sample ChemicalTagger markup of a reactant

The quantities of a MOLECULE may be a MASS, AMOUNT or VOLUME. These elements occur either as a child of a QUANTITY element, as seen above, or of a MIXTURE element if further text content is present, *e.g.* “4-(dimethylamino)-benzenethiol (.147g, .96mmol, prepared in step 2)”;

```

<MOLECULE>
  <OSCAR-CM>4-(dimethylamino)-benzenethiol</OSCAR-CM>
  <MIXTURE>
    <_LRB-> (</_LRB->
    <MASS>
      <CD>147</CD>
      <NN-MASS>g</NN-MASS>
    </MASS>
    <COMMA>,</COMMA>
    <AMOUNT>
      <CD>96</CD>
      <NN-AMOUNT>mmol</NN-AMOUNT>
    </AMOUNT>
    <COMMA>,</COMMA>
    <VB-SYNTHESIZE>prepared</VB-SYNTHESIZE>
    <IN-IN>in</IN-IN>
    <NN>step</NN>
    <CD>2</CD>
    <_RRB->)</_RRB->
  </MIXTURE>
</MOLECULE>

```

Figure 5-22: ChemicalTagger output for mixed content

It is presumed that any `MOLECULE` that occurs in the synthesis text and has a `MASS`, `QUANTITY` or `VOLUME` is a reagent. CML representations including connection tables are generated for reagents identified in this way by using the `NameResolver` class of OSCAR3 – which was modified in order to create the capacity to access its functionality from external applications. A solvent is then distinguished by one of two criteria – that it is a member of a list of known solvents, or that the quantity quoted for the `MOLECULE` is given only as a volume. In order to avoid the necessity of creating a large dictionary of synonyms for common solvents, *e.g.* “DCM”, “dichloromethane” or “methylene chloride”, the check against the list of known solvents is performed by generating from connection table an InChI using the `InChIGeneratorTool` class in JUMBO and checking this InChI against a known list.

The reagents identified in this way are then used to populate the `ReactantList` and `SpectatorList` children of the CML Reaction. For example, the lists in Figure 5-23 are generated for the following example;

Into a round bottom flask fitted with a gas inlet, condenser and a magnetic stirring bar was placed methyl 8-(benzyloxy)-5-bromo-1,6-naphthyridine-7-carboxylate (.4 g, 1.03 mmol), 2,3-dimethoxybenzylamine (.432 g, .38 mL, 2.58 mmol) and 10 mL toluene: This mixture was refluxed for 18 hours, after which the reaction was cooled and the solvent removed in vacuo. The resulting residue was triturated with diethyl ether and filtered to yield 5-bromo-N-(2,3-dimethoxybenzyl)-8-hydroxy-1,6-naphthyridine-7-carboxamide as a light yellow solid.

```

<reactantList xmlns="http://www.xml-cml.org/schema">
  <reactant>
    <molecule id="m1"
      title="methyl 8-(benzoyloxy)-5-bromo-1,6-naphthyridine-7-
      carboxylate">
      <identifier convention="iupac:inchi">
        InChI=1/C17H11BrN2O4/c1-23-17(22)13-14(24-16(21)10-6-3-
        2-4-7-10)12-11(15(18)20-13)8-5-9-19-12/h2-9H,1H3
      </identifier>
      <atomArray>
        ...
      </atomArray>
      <bondArray>
        ...
      </bondArray>
    </molecule>
    <amount units="cml:g">0.4</amount>
    <amount units="cml:mmol">1.03</amount>
  </reactant>
  <reactant>
    <molecule id="m1" title="2,3-dimethoxybenzylamine">
      <identifier convention="iupac:inchi">
        InChI=1/C9H13NO2/c1-11-8-5-3-4-7(6-10)9(8)12-2/h3-
        5H,6,10H2,1-2H3
      </identifier>
      <atomArray>
        ...
      </atomArray>
      <bondArray>
        ...
      </bondArray>
    </molecule>
    <amount units="cml:g">0.432</amount>
    <amount units="cml:mmol">2.58</amount>
    <amount units="cml:mL">0.38</amount>
  </reactant>
</reactantList>
<spectatorList>
  <spectator role="solvent" hasOnlyVolume="true">
    <molecule xmlns:cmlx="http://www.xml-cml.org/schema/cmlx"
      cmlx:explicitHydrogens="true" title="toluene">
      <atomArray>
        ...
      </atomArray>
      <bondArray>
        ...
      </bondArray>
    </molecule>
    <amount units="cml:mL">10.0</amount>
  </spectator>
</spectatorList>

```

Figure 5-23: Automatically generated reactantList and spectatorList. For the sake of brevity, atom and bond elements have been removed

It can be seen that, using the techniques discussed above, the software has correctly identified the reactants (methyl 8-(benzoyloxy)-5-bromo-1,6-naphthyridine-7-carboxylate and 2,3-dimethoxybenzylamine), the solvent (toluene), and the quantities of each that were used.

This approach, however, is only applicable to syntheses that are described in a straightforward manner, *i.e.* those that directly identify the reagents employed and do so using interpretable nomenclature. Those that require the addition of information from other sections of the document by use of the previously discussed back reference require alternative techniques to be employed.

These references, where identified, have by this stage in the workflow been annotated (see section 5.2.6). The treatment of reactions that are described by analogy to another synthesis is an involved process that is described in full later, while syntheses that use compound references are resolved by direct text substitution of the reference text with the name of the compound to which the reference refers prior to running ChemicalTagger over the text. For the example;

```
Triphosgene (0.556g, 1.87 mmol) was added over 20 mins to a
solution of
  <compoundRef targetId="h0012">
    the acid from step 1.
  </compoundRef>
(0.89g, 4.68 mmol) and diisopropylethylamine (3.26 ml, 18.7
mmol) in DMF (22 ml) at 0°C.
```

The identity of “the acid from step 1” is determined by examining the referenced section of document. It is assumed that such references refer to the product of the referenced reaction rather than to any other compound involved. It would be preferable, in cases such as the above where the referenced compound could indeed be some compound other than the product, for the appropriate compound to be selected from the list of those compounds involved in the reaction in order to ensure that this assumption is correct. The product of the referenced reaction is determined in the same manner as described previously, by passing the title text of the heading to which the `compoundRef` element points to OSCAR3 in order to identify the name of the target compound. This

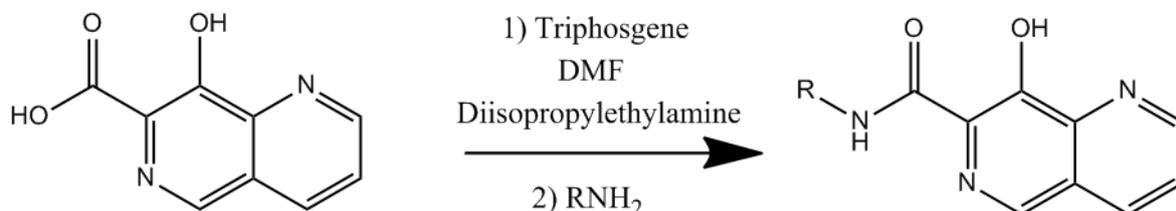
compound name is then added to the subject text in place of the compound reference, which for the above example yields;

Triphosgene (0.556g, 1.87 mmol) was added over 20 mins to a solution of 8-hydroxy-1,6-naphthyridine-7-carboxylic acid (0.89g, 4.68 mmol) and diisopropylethylamine (3.26 ml, 18.7 mmol) in DMF (22 ml) at 0°C.

Which is a format from which the reagents may be correctly identified using the previously described method.

5.3.2.4 Resolution of Analogous Reactions

The method for identifying reagents described in section 5.3.2.3 is, of course, dependent upon the patent author directly describing the synthesis, as opposed to defining it by analogy to a previous reaction. The patent EP1326865 contains a number of reactions of the form;



This reaction is fully described once; in Example 2, Step 2 in which RNH₂ is 2,5-dichlorobenzylamine.

Subsequent analogous reactions may be described in such a manner as;

The title compound was prepared using the procedure described in Example 2, Step 2 replacing 2,5 dichlorobenzylamine with 1(R,S) aminoindane.

or;

The title compound was prepared using the procedure described in Example 2, Step 2 from 8-hydroxy-1,6-naphthyridine-7-carboxylic acid and 1(R,S) aminoindane.

The first of these formulations explicitly instructs the reader which of the reagents from the reference reaction is to be replaced, while the second does not. Examples of the first formulation could be dealt with to an acceptable degree of success by matching phrases of the form “replacing/substituting/replacement of/substitution of X for/with Y” *etc.* (where X and Y are chemical names) and modifying the CML Reaction produced from the source reaction accordingly. This approach, however, would not be helpful in the resolution of examples of the second formulation. Since the text does not inform the reader which of the reagents was replaced by the aminoindane, it is necessary to apply a degree of chemical knowledge in order for it to be determined. The algorithm implemented to resolve this problem is similar to the mental process that a chemist might use.

First, consider the reagents employed and product produced in the reference reaction. A trained chemist, even without applying any knowledge of the reactivities of the species involved, will be trivially able to identify that a coupling has occurred between the carboxylic acid and the amine. This assertion may be made based on the high degree of common substructure between the two reagents and the product, and is illustrated in Figure 5-24.

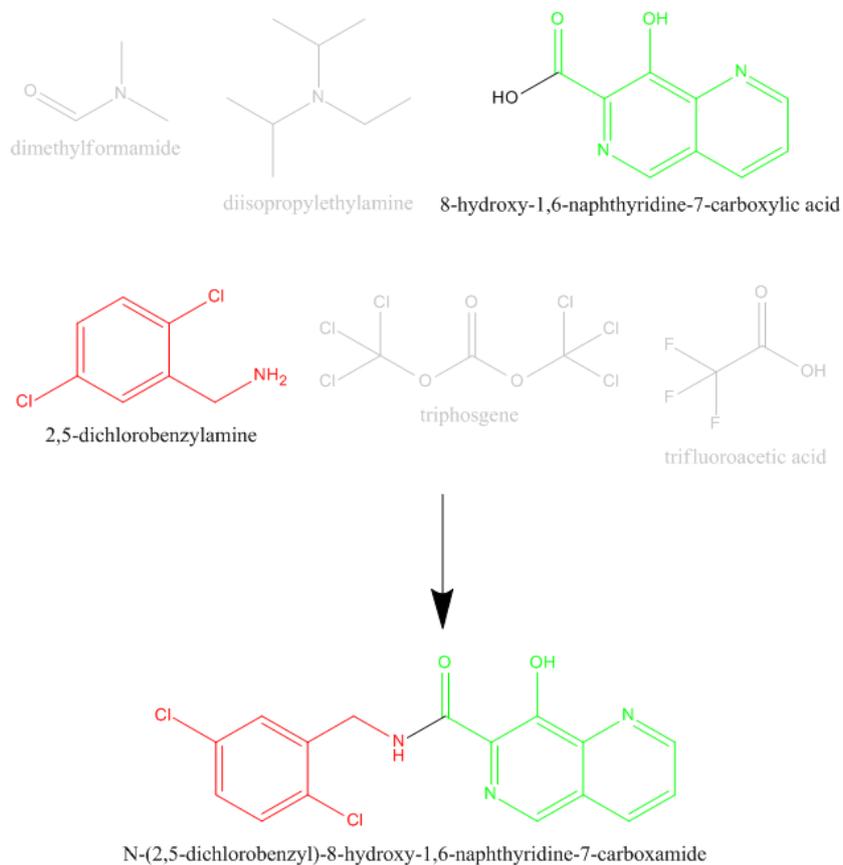


Figure 5-24: Identification of key reactants

The analogous reaction tells us that the product, in this case N-[(1R,S)-2,3-dihydro-1H-inden-1-yl]-8-hydroxy-1,6-naphthyridine-7-carboxamide, is prepared “from 8-hydroxy-1,6-naphthyridine-7-carboxylic acid and 1(R,S) aminoindane”. We thus know that the carboxylic acid and the aminoindane are employed in the analogous reaction, and by considering which of the reagents share a significant substructure with the product, we observe the following;

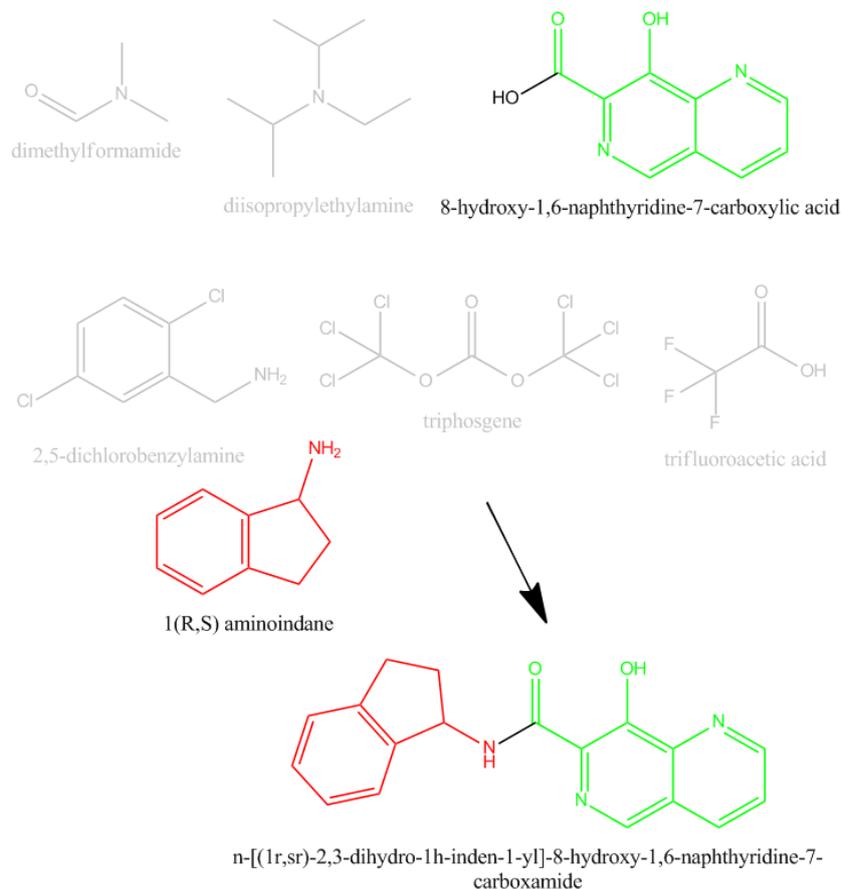


Figure 5-25: Significant substructures in the analogous reaction

The product of the reference reaction is composed of substructures derived from the dichlorobenzylamine and the carboxylic acid, while that of the product of the analogous reaction is composed of substructures derived from the aminoindane and the carboxylic acid. It may therefore be concluded, since the dichlorobenzylamine played a role in the reference reaction that it does not in the analogous reaction, that the dichlorobenzylamine is not used as a reagent in the analogous reaction.

This procedure is implemented as follows;

- First, the reference reaction is analysed by the ReactionMapper class. The CML Reaction that was generated for the reference reaction states the product of the reaction and the reagents from which it is synthesised. For each of the reagents in turn, the maximum

common substructure with the product molecule is calculated using the Chemix library (108), employing a bond-directed search and disregarding hydrogen atoms. From these is determined the smallest set of reagents which between them map to all of the non-hydrogen atoms of the product molecule. This replicates the mental procedure illustrated in Figure 5-24.

- Chemicals named in the description of the analogous reaction are identified by OSCAR3. These names are resolved to connection tables using OSCAR3's NameResolver class and, if necessary, converted to CML. These CML Molecules, along with the ReactionMapper from the previous step and the CML Molecule for the new product of the analogous reaction (determined as in section 5.3.2.1), are passed to the ModifiedReactionMapper class. This class creates a new ReactionMapper for the new reaction and provides the additional functionality to resolve the analogous reaction while delegating previously implemented functions to its new ReactionMapper. Each of the reagents identified as being part of the smallest complete mapping set in the previous step has their maximum common substructure with the new product molecule determined as before, as do each of the molecules mentioned in the reaction description of the analogous reaction. The smallest complete mapping set for the analogous reaction may then be determined as before. It is then possible to identify the replaced reagents as being those that were members of the first smallest complete mapping set but not of the second.

A CML Reaction for the analogous reaction is then compiled. The CML Reaction for the reference reaction is copied and updated to reflect the differences between the two reactions. The product molecule of the original reaction is replaced with the new product molecule, while the new reagent molecules that are mentioned in the description of the analogous reaction and contained within the

smallest complete mapping set are added to the `reactantList` and those identified as not occurring in the analogous reaction by the `ModifiedReactionMapper` are removed.

This method of resolving analogous reactions is suitable for application to both reaction descriptions that make explicit statements of a reagent to be replaced and to those that rely on the reader to be able to infer this information. When a reagent that is already present in the smallest complete mapping set for the reference reaction is again mentioned in the description of the analogous reaction, such as the carboxylic acid in the previous examples, it will be duplicated and passed to the `ModifiedReactionMapper` as though it were a novel reagent. Since only one of the instances of the molecule is required in the `ModifiedReactionMapper`'s smallest complete mapping set, however, it can be guaranteed that only one of the copies of the molecule will occur in the CML Reaction produced for the analogous reaction.

5.3.2.5 Automated Verification of Product

Of course, it cannot be guaranteed that the connection table generated for the product molecule is correct. There are several potential sources of error – the name used in the heading in the original patent document may have been incorrect, the wrong heading may have been associated with the experimental section being processed, OSCAR3 may have misidentified a chemical name *e.g.* it may have annotated only “ethanoic acid” in the string “ethanoic acid methyl ester” or the connection table may have been incorrectly generated from the chemical name. As a result, it is desirable to be able to automatically verify the product in some way. This can be achieved by comparing the determined product to the extracted spectral data and, if present, any accompanying chemical images. The process of acquiring these sources of information must also be regarded as potentially inaccurate, and so it is not possible to definitively confirm or refute any candidate product.

Nonetheless, these checks provide potentially useful information regarding the validity of the assigned product and of the assigned spectral data.

Checking Against ^1H -NMR

Given the ^1H NMR spectrum of an unknown compound, it is possible for one skilled in the art to discount certain candidate structures. Most trivially, the proton count in the candidate structure should agree with total integral of the NMR spectrum. Each unique chemical environment in the candidate structure should give rise to a distinct peak in the NMR spectrum and it should be possible to assign for each of the chemical environments a peak that is in the correct region of the NMR spectrum. The peak multiplicities should be explained by potential couplings in the candidate molecule, and protons that couple to one another should share coupling constants.

The application of these rules is subject to a large amount of subtlety, however. While each chemical environment should give rise to an individual peak, these peaks can overlap and be indistinguishable from one another, most notably in the case of aromatic protons. The determination of unique chemical environments is complicated by the need to consider three dimensional effects, as illustrated in Figure 5-26. Here, a plane of symmetry defined by R, H_a and the carbon atom that connects them exists through the molecule. Thus the protons H_b are equivalent with one another but not with their geminal protons H_c , which sit on the other face of the ring, made inequivalent by the group R.

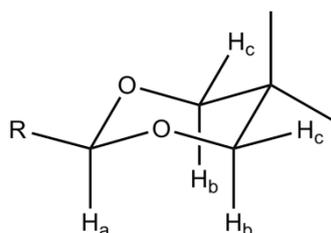


Figure 5-26: Proton environments in a non-trivial system

As a result of these effects, it is not possible to compute chemical environments based solely on the 2D connectivity of a molecule and confidently assert that this will equal the number of peaks reported in the molecule's NMR spectrum. Whether there are more or fewer peaks in the NMR spectrum than predicted, the candidate molecule may be correct. Conversely, if the prediction matches the observation the candidate molecule may still be incorrect. The resolution of this problem falls outside of the scope of this project, and so the checking of structures against ^1H NMR spectra is limited to the first method mentioned – ensuring that the proton count of the molecule agrees with the total integral of the spectrum. The result of this check is recorded in automatically generated CML Reaction by adding a `matchesHnmr` attribute to the product molecule, with a value of true or false as appropriate.

Checking Against Mass Spectrum

As described in section 5.2.3.3, the regular expressions used by OSCAR3 to annotate mass spectra were developed in order to facilitate the specific annotation of peak assignments. Coupled with the reported mass, this provides the information required to check the candidate product against the reported mass spectrum. In this context, assignments are given as an offset from the molecular ion, *e.g.* “M+1” or “M–H”. The functionality to parse the assignment text and calculate the mass offset it implies is implemented in the `MassSpecAssignment` class, which supports both the numeric offset and formula offsets illustrated above. Regular expression matching determines the type of offset employed, or lack of an offset, *e.g.* “m/z: 271 (M⁺)”. Where a formula offset is used, the mass of the corresponding molecular fragment is determined using the `CMLFormula` class in CMLXOM.

The reported mass is annotated as a numeric value by OSCAR3, and so once the mass offset has been determined it may be trivially added or subtracted from the mass of the reported peak to calculate an expected mass for the candidate product molecule. This mass is not necessarily the same as the average molecular mass of the product, since in mass spectrometry it is the individual

masses of the isotopomers that is measured. The discrepancy between the two would be of critical importance when considering, for example, any chlorine-containing product, and so it is necessary instead to be able to determine the masses of the isotopomers of a given molecule. This functionality was not previously available in JUMBO, and so was implemented and added to the `MoleculeTool` class for this purpose.

The result of comparing the calculated mass of the reported molecular ion with the calculated mass of the most abundant isotopomer is recorded by adding a `matchesMassSpec` attribute to the product `molecule` in the automatically produced CML Reaction, with a value of true or false as appropriate.

Checking Against Embedded Images

When the experimental section includes a chemical image it is possible to compare the connection table of the candidate product with the results from the OSRA analysis of that image, which by this stage in the workflow have been appended to the patent document XML (see section 5.2.5). If a chemical image is found within the source experimental section, the recorded SMILES strings for the image are built into CML Molecules which are then used to generate InChIs, using the `SMILESTool` and `InChIGeneratorTool` classes from JUMBO respectively. An InChI for the candidate product molecule is similarly generated, and the InChIs are subsequently compared.

Since the image included in the experimental section may be a reaction diagram it is possible for OSRA to have identified more than one molecule. Since the analysis of the often low-quality images is an error prone procedure, it is possible that the structures identified in the image may not contain the correct product. As previously discussed, when OSRA fails to correctly deduce a connection table from a drawn structure, it frequently reports a result containing the wildcard character, "*". This character is not recognised by JUMBO's `SMILESTool`, causing it to throw an `Exception`. As a

result, the following rules are applied when checking the candidate product against embedded images;

1. If the InChI generated for the candidate product matches one generated for the structures identified by OSRA, the product is considered to match the image.
2. If all of the structures identified by OSRA can be built into InChIs and the InChI for the candidate product does not match one of these, the product is considered not to match the image.
3. If some or all of the structures identified by OSRA cannot be built into InChIs and the InChI for the candidate product is not matched by one of those generated from the chemical image, no conclusion is drawn.

If a conclusion is drawn from this process, it is recorded in the CML Reaction by the addition of a `matchesImage` attribute on the product CML Molecule, with a value of true or false as appropriate. If no conclusion is drawn, or if the source experimental section does not contain a chemical image, the CML Reaction is not modified.

5.3.3 Reaction Extraction Performance

As previously described, the inputs to the reaction extraction procedure are those sections of the semantically enhanced patent documents that are expected to describe experimental procedures. XPath is used to select the appropriate headings from the patent, and the section of the document that is subjected to the analysis is this heading along with all of its descendant nodes. A successful analysis of the experimental section results in a CML Reaction, as discussed in section 5.3.2.

It is not the case, however, that the input of each and every one of the sections of the document that are used as an input will successfully result in a CML Reaction. The reaction extraction

procedure may fail to generate a CML Reaction for a number of reasons. These causes were assessed by monitoring the results of the procedure when run on the 26287 input sections selected from the set of semantically enhanced patent documents. From these input sections, 4444 CML reactions were successfully extracted, representing around 17% of the total input. The major causes of failure are subsequently discussed;

- No text-containing paragraph children – it is expected that text should be contained inside `p` elements; if the input section does not contain recognisable text and does not, therefore, describe an experimental procedure in a manner that the system is designed to handle then it does not attempt to produce a CML Reaction. This situation arises where the restructuring of the patent document XML has not correctly organised the complete description of an experimental procedure into a single section of the semantically enhanced XML document, and was observed on 2595 occasions – around 10% of the reaction extraction attempts.
- Too many paragraph children – if the input section has more than one text-containing paragraph child, then the system does not attempt to process it. This strategy is employed since a situation where one heading has multiple paragraph children may describe a single-step synthesis split across the paragraphs or may describe a multi-step synthesis where each paragraph describes a different reaction. In this second scenario, the techniques currently used to abstract the reaction would be inappropriate and so in the absence of an available method to distinguish between the two cases, it is impossible to determine how to proceed. Recent developments in the ChemicalTagger project that classify the role of each phrase in an experimental description, for example to produce “dissolve”, “heat” and “yield” phrases, may provide a means by which the paragraphs may be determined to describe either a single or multiple synthesis though this has not been implemented within the current work. This situation was observed on 2747 occasions, or around 10% of the total.

- No product identified – the system attempts to determine the product by using OSCAR3 to recognise a single chemical name in the heading of the experimental section. If this process cannot be successfully completed, whether because the author has not named a single chemical compound in the heading or because OSCAR3 has failed to correctly identify the chemical name therein, the process aborts. This situation was observed on 5236 occasions, or around 20% of the total.
- No reagents identified – if the system fails to identify any reagents in the source text, *i.e.* if there are no chemical names with associated amounts as recognised by ChemicalTagger, then no CML Reaction is produced. This situation may occur if the input describes an analogous reaction in which the backreference has not been annotated, if the patent author described the amounts of reactants in a format which is not supported by ChemicalTagger or if the input text does not describe a reaction and therefore does not describe amounts of chemicals. This situation was observed on 2562 occasions, or around 10% of the total.

Further to the causes examined and discussed above, the failure to resolve backreferenced reactions caused a significant number of failures. These failures occurred primarily because the reference reaction had not been successfully extracted, because the extracted reference reaction failed to correctly identify and resolve to a structure all of the reagents required for the resolution procedure to succeed, or because the maximum common substructure searching element of the resolution procedure timed out. Though these errors were not automatically categorised, they are believed to have occurred for around 30% of the inputs.

5.4 Conclusions

The work in this chapter has demonstrated the potential to create a system that downloads and processes the chemical literature and extracts machine-understandable data from it without the need for user interaction. The resulting CML reactions are further described in the next chapter.

Much of the software that comprises PatentEye is likely to be of use to related projects in the future. Though the code is not currently modularised, it has already seen reuse as part of the Green Chain Reaction (109) – a distributed experiment to answer the question “are chemical reactions in the literature getting greener?” The Green Chain Reaction reused the PatentEye code for the identification, downloading and semantic enhancement of chemical patents and demonstrated the value of the work described in this chapter – as well as suggesting areas in which the PatentEye code and some of the libraries upon which it depends would benefit from refactoring. Since the community is unlikely to adopt software tools that are not both robust and usable, this exercise has proved valuable and it is hoped that the points raised will be addressed in the future.

6. Results

The previous chapter described the means by which it was possible to accomplish high-throughput extraction of reactions from the patent literature. This chapter discusses the analysis of these extracted reactions to assess the performance of the PatentEye system (section 6.1) and briefly discusses the work of Dr Lezan Hawizy to enable the reuse of the results of the current work by third-parties (section 6.2).

6.1 Quality of Extracted Reactions

To assess the accuracy of the semantified reactions, the output of the reaction extraction process was manually examined and compared with the source text. Each CML reaction was assessed on a number of criteria to determine the performance of the different modules of the reaction extractions system. These criteria included the accuracy of identified products, reagents and spectra, and the performance of the systems for automated product verification was tested by comparing the results of the automated verification with those of the manual verification. The methods employed for this process and the results obtained are subsequently discussed.

6.1.1 Corpus Formation

Since the manual inspection of each and every reaction extracted from the patent texts was not a feasible task, a subset was selected to serve as a corpus from which to derive performance metrics. From the 4444 reactions successfully extracted from the 667 unique, full-text patent documents, 100 reactions were selected at random. This reaction corpus was then used in the subsequently described validation procedures.

During the manual inspection of the reaction corpus, it was discovered that two of the 100 CML Reactions were derived from multi-step syntheses that were described within a single paragraph. Since these cases did not reflect the kind of input for which the current software was designed, as discussed in section 5.3.3, they were excluded from the analysis process. A further two CML Reactions in the reaction corpus were found to have been derived from examples of their respective inventions that did not describe chemical syntheses – instead describing assays. These CML Reactions were similarly excluded from the analysis process; consequently, the process is based upon a reduced corpus of 96 CML Reactions.

6.1.2 Product Validation

The source from which the reaction was extracted was examined to determine whether the chemical name identified in the heading text by OSCAR3 and from which the product CML molecule was generated agreed with that stated in the heading text. Since the name to structure conversion process is not a perfect procedure, this is no guarantee that the attached connection table is also correct. However, the development of OPSIN was not a part of the current work and is reported to operate at an extremely high rate of performance (110) and so it was not considered necessary to measure the accuracy of this process. Each extracted reaction is required by the extraction process to contain one and only one product molecule and so the product validation for each reaction was recorded as a simple boolean.

The manual inspection of the reaction corpus showed that the correct product was identified on 88 of the 96 occasions, a success rate of around 92%. It was further noted that on each of the 8 occasions on which the correct product was not identified, the term identified as the product name could not be successfully resolved to a connection table, suggesting a means by which the errors may be automatically removed. Generally, the cause of the failure to identify the correct product

was due to the product of the reaction being named in the accompanying text, and hence not being present in the section heading of the source; instead, a term from the heading was falsely identified as a chemical name, which allowed for the creation of a CML Reaction from the source.

The high rate of successful product identification suggests that the current techniques are sufficient to extract high-quality data from the literature in this regard. Those errors in product identification that exist at the present time may be largely eliminated in the future by the application of NLP techniques to permit the identification of products from the reaction description text.

6.1.3 Reagent Validation

The sources from which the reaction corpus was extracted were examined, and for each the reagents employed and the amounts thereof were identified. These were then compared with those automatically extracted; instances where the same chemical name and amount were both manually identified and automatically extracted were counted as true positives, where the automatically extracted reagent list contained an instance that was not matched by both chemical name and amount in those manually identified a false positive was counted, and where a reagent was manually identified that was not automatically extracted, a false negative was counted.

This work required the formalisation of the concept of a reagent to a sufficient degree that any subjectivity in determining what did and did not constitute a reagent could as far as possible be minimised. The IUPAC definition, “a test substance that is added to a system in order to bring about a reaction or to see whether a reaction occurs” (107), does not match the common usage of the term which further includes the chemical species involved in a reaction, *i.e.* reactants, solvents, catalysts, *etc.* It is this wider definition that fits the goal of the current work – to automatically determine how a reaction is carried out.

It was observed when considering this task that the chemical literature frequently underspecifies the work-up stage of a reaction. That is to say, the reagents employed may be stated without reference to their amounts, such as in;

“The reaction mixture was stirred at 25 °C for 4 days and then diluted with ethyl acetate. The mixture was then washed with a dilute aqueous hydrochloric acid solution. At this time, methanol was added to the organic layer. A precipitate formed and was removed by filtration. The organics were further washed with a saturated aqueous sodium chloride solution, dried over magnesium sulfate, filtered, and concentrated in vacuo. The resulting solid was triturated with diethyl ether. The solid was collected by filtration and washed again with diethyl ether to afford...” (111)

While the work-up is an undeniably important phase of a reaction, the techniques used in the current work are reliant on the specification of amounts in order to identify reagents. This technique is well-suited to identification of primary reagents but not those used in work-up, and so in order to produce a metric that indicates the performance of the software in the role for which it was designed it was decided to entirely omit reagents mentioned in the work-up phase, and inert atmospheres under which reactions were performed, from the current analysis.

The manual inspection of the reaction corpus identified 249 true positives, 71 false positives and 139 false negatives – the system having a precision of around 78% and recall of around 64%. When considering these results, it should be remembered that the requirement for an identified reagent to be considered a true positive – that not only the chemical name but also the amounts employed in the reaction be identical to those described in the source text – is a rigorous standard. It was commonly the case during the analysis that the system identified the correct chemical name as a reagent but failed to correctly add one or more amounts, creating both a false positive and a false negative. These situations occurred where one or more of the amounts in the source text were not recognised by the regular grammar employed by ChemicalTagger for amount recognition. Frequently these situations were caused by the patent author employing a structure that may be

considered incorrect, *e.g.* “triphenylphosphine (3.08g., 11.78 mmol)” or “1-Phenylpiperazine (16.2 g, 0.10 mole)”. The non-standard full stop indicating the abbreviation of “grams” in the first example and the failure to contract the unit “mole” to its standard symbol “mol” in the second result in the failure to recognise and convert these amounts to CML. The data gathered in the current exercise permit the improvement of the ChemicalTagger grammar to recognise a greater variety of the reporting formats used by authors and thereby improve the precision and recall for the identification of reagents as measured by the current methods.

These improvements, however, are not sufficient on their own to produce a system that operates at the level of a human operator. The current system requires further development before the data it produces are of sufficient quality to be considered reliable by the community at large.

6.1.4 Spectra Validation

The extracted reactions contain, where identified and successfully converted to CML, the ^{13}C and ^1H NMR spectra of the products. In the patents used for this work, ^1H NMR spectra are far more common than ^{13}C (see section 5.2.3.4) – indeed, the manually examined subset of the reaction corpus was found to contain only two ^{13}C NMR spectra. Consequently, only the validity of the attached ^1H NMR spectra in the reaction corpus was considered. Where these spectra were present, the content was compared to the reported spectra in the original sources. In order to be considered correct, the attached spectra were required to fully describe the original spectra in terms of the shifts, integrals, multiplicities and coupling constants of each peak – any deviation from what was reported in the original text resulted in the attached spectrum being judged to be incorrect.

The manual inspection identified 25 occasions on which the ^1H NMR spectrum attached to a product molecule precisely replicated the information presented in the source text and 8 occasions on which it did not, *i.e.* a success rate of around 76%. The primary causes of the inclusion of incorrect ^1H NMR

spectra were the failure to fully convert peak metadata, *e.g.* multiplicities, as identified by OSCAR3 to CML and the conversion to CML spectra of sections of input text that did not indicate ^1H NMR spectra, *i.e.* false positives in the data recognition procedure. The first of these issues indicates a bug in JUMBOConverters that could be relatively trivially identified and fixed while it is expected that the second issue should produce ^1H NMR that could be automatically distinguished from a genuine NMR spectrum in a majority of cases, since false positives will rarely contain expected peak metadata such as integrals and multiplicities. This was not, however, attempted in the current work since the identified sample was so small as to be statistically meaningless.

Though the ^1H NMR spectra validation is based on a small set of data, it is believed that the spectra identified by PatentEye are of nearly sufficient quality that they constitute a resource of value to the community.

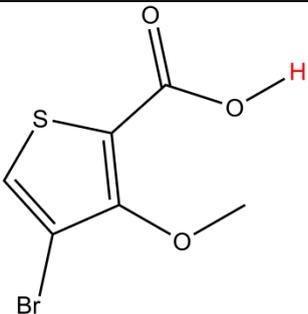
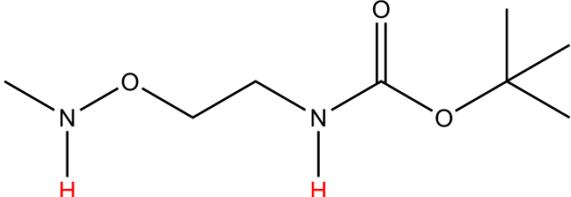
6.1.5 Automated Verification Validation

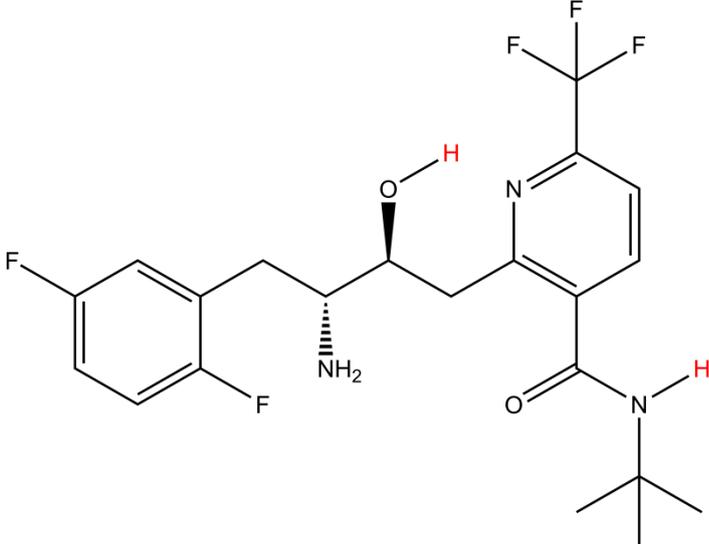
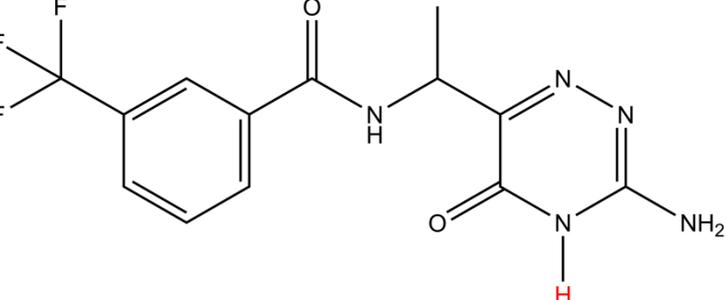
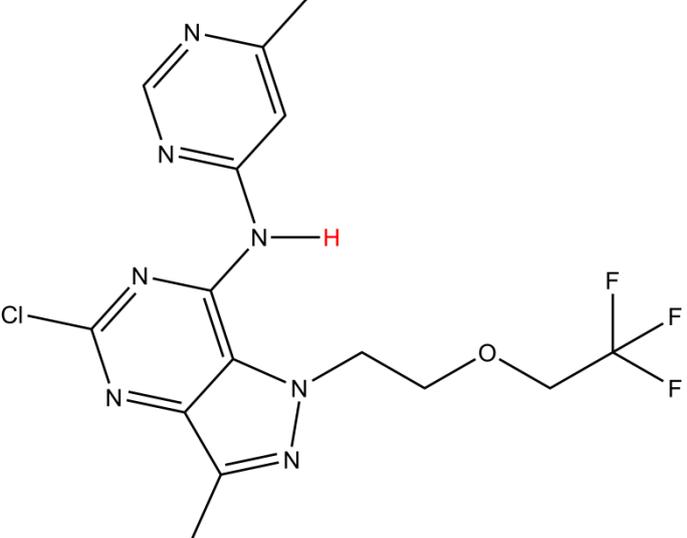
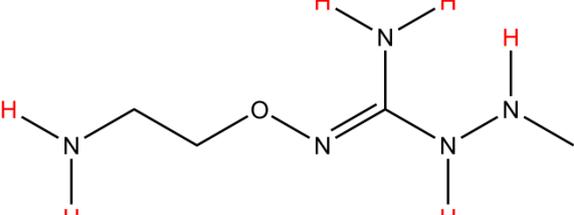
For each extracted reaction in the corpus, the results of the automated product verification, as described in 5.3.2.5, were compared with those of the manual product validation. For each of the methods of automated product verification – comparison of the product structure with chemical images, with its mass spectrum and with its ^1H NMR – the automated verification was judged to be correct where the automated and manual verifications were in agreement and judged to be incorrect where they were not. The results of this examination of the reaction corpus are subsequently discussed.

6.1.5.1 ¹H-NMR Verification

As discussed in section 5.3.2.5, the verification of products using their ¹H NMR spectra is performed by matching the proton count of the candidate product molecule to the total integral of their respective ¹H NMR spectrum. Of the 33 products in the reaction corpus with attached ¹H NMR spectra, 23 (70%) were considered to have matched the product molecule while 10 (30%) were not. In all of the 33 cases, the product was manually judged to have been identified correctly.

The 10 cases in which the extracted ¹H NMR spectrum was judged not to match the candidate product molecule were examined to determine the cause. In one of these cases, the spectrum had been derived from a false positive in the OSCAR3 data recognition process. In the remaining nine cases, the total integral of the recorded spectrum was less than would be expected for the candidate product molecule. This is generally presumed to have been a consequence of the method by which the spectrum was acquired rather than indicative of an incorrect spectrum or mistaken product. The nine candidate product molecules and the solvents in which the spectra were reported to have been acquired are shown in Table 6-1.

Candidate Product	Solvent
	CDCl ₃
	CDCl ₃

	<p>CDCl₃</p>
	<p>DMSO-d₆</p>
	<p>DMSO-d₆</p>
	<p>D₂O</p>

	CD ₃ OD
	CD ₃ OD
	CD ₃ OD

Table 6-1: Candidate product molecules for which the ¹H NMR was considered a non-match, with likely missing hydrogen atoms indicated in red

For each of the candidate product molecules in Table 6-1, the n most acidic protons, where n is the difference between the proton count of the candidate product molecule and the total integral of the ¹H NMR spectrum, are shown in red. It is supposed that in each case, those protons were either dissociated or exchanged with acidic deuterons from the solvent and so were not reflected in the observed spectrum. This demonstrates the fragility of the chosen method for validation of products

via their ^1H NMR spectra, and so it is recommended that in the future a more advanced method of structure-spectrum comparison be employed.

6.1.5.2 Mass Spectrum Verification

Of the 24 products in the reaction corpus for which automated verification against a mass spectrum was attempted, 18 (75%) were considered to have matched the candidate product molecule while 6 (25%) were not. In all of the 24 cases, the product was manually judged to have been identified correctly. The cases in which the reported mass spectrum was judged not to be a match to the candidate product molecule represent a combination of cases in which the OSCAR3 methodology was insufficient to deal with the reported mass spectrum, *e.g.* “ m/z : 594.9 (M+H-HCl)” and “MS m/z 471, 473 (M+1, M+3)” and those in which the reported spectrum and assignment is apparently wrong, *e.g.* for the case of 2-fluoro-4-(4-fluoro-benzyloxy)-nitrobenzene (where the mass of the most commoner isotopomer is 265.1), the spectrum was reported as “MS: $m/e = 265.1$ (M⁺+H)”.

6.1.5.3 Image Verification

Of the 26 products in the reaction corpus for which automated verification against an attached image was attempted, 8 (31%) were considered to have matched the candidate product molecule while 18 (69%) were not. In all of the 26 cases, the product was manually judged to have been identified correctly. That the automated verification of candidate product molecules against attached images is highly error-prone should not be surprising, since the performance of the chemical image recognition package employed in the current work has previously been shown to perform poorly on a corpus of images taken from EPO patents (see section 5.2.5.2).

6.1.5.4 Conclusions

Given the high accuracy (92%) of the PatentEye system in terms of correctly identifying the product of a chemical reaction compared to the accuracy of the implemented systems of automated product verification, it may be considered that this verification process is at best unnecessary and should be discontinued in future versions of the PatentEye system. However, since the NMR spectra reported in the literature are themselves data that are likely to be of interest and of use to the community, it is desirable to continue to collect and to validate these. The current, naïve, system of validation has been shown to be insufficient for the purpose and so a more capable method must be developed for the future. It is envisioned that such a system would operate in two parts; a “sanity check” would identify those spectra that are clearly in error and included as a result of a false positive in the OSCAR3 data recognition, *e.g.* those that occur in the middle of a chemical name, before a chemically intelligent system ensures that the extracted spectrum could feasibly correspond to the candidate product molecule under the specified conditions. Such a system would be capable of fully-automatic production of high volumes of reliable data to be distributed to the community.

6.2 Enabling Reuse of the Extracted Data

This thesis began by discussing the need for machine-understandable and open data. The work described up to this point has shown how it is possible to extract machine-understandable data from the chemical literature, but the issue of how this data may be disseminated in a way that supports the semantic web of chemistry has not been addressed. Of course, the XML files produced as part of the current work could be hosted on a webserver from where they could be downloaded by interested members of the community. This approach, however, neglects to provide interoperability between the data produced by PatentEye and that produced by any other source. In this scenario, drawing

together data from different sources is a problem that would need to be solved and re-solved by any number of different users.

The issue of how best to produce *linked data* is addressed by the Resource Description Framework (RDF) (112), a web-standard technology for the encoding of knowledge. In RDF, statements about resources (concepts) are made using the subject-predicate-object format, *e.g.* “Marlon Brando starred in The Godfather”, in which the subject is “Marlon Brando”, the predicate is “starred in” and the object is “The Godfather”. Resources, such as “Marlon Brando” and “The Godfather” are defined by Uniform Resource Indicators (URIs), allowing other authors of RDF to make further statements about the same resources such as “Marlon Brando married Anna Kashfi” or “Al Pacino starred in The Godfather” and for the knowledge represented by the statements to be automatically combined – provided that the authors use the same URIs to define the same resources. Using RDF for the communication of chemical knowledge has the additional advantage that a host of tools for its usage exist, such as the open-source framework Sesame (113) and the query language SPARQL (114).

This conversion of the information extracted from the chemical literature to RDF has been addressed by Dr Lezan Hawizy, with the creation of the PatentEye Repository (115). In the RDF files upon which the PatentEye Repository is based, the reactions extracted from the literature in section 5.3 are combined with the molecular classifications derived in section 4.3. Each instance of each chemical from the reactions is represented by a separate resource. Each of these resources links to a resource that defines the molecule concerned, thus allowing the different instances of the same chemical from across the literature to be drawn together and to be combined with data from other sources. The URIs for the resources used to define the molecules are provided by the Chemical RDF project hosted by OpenMolecules (116) and are of the form <http://rdf.openmolecules.net/?XXX> where XXX is the InChI of the molecule concerned. Some example RDF from the PatentEye Repository is shown in Figure 6-1, and the structure of this data is illustrated in Figure 6-2.

```

<rdf:Description rdf:about="http://www.patenteye.com/#ep01778686b1">
  <j.0:hasReaction
    rdf:resource="http://www.patenteye.com/#ep01778686b1-h0023" />
  <rdf:type rdf:resource="http://www.patenteye.com/#Patent" />
</rdf:Description>
<rdf:Description
  rdf:about="http://www.patenteye.com/#ep01778686b1-h0023">
  <rdf:type rdf:resource="http://www.patenteye.com/#Reaction" />
  <j.0:hasMolecule rdf:resource=
    "http://www.patenteye.com/#ep01778686b1h0023isopropylalcohol2
    00" />
</rdf:Description>
<rdf:Description rdf:about=
  "http://www.patenteye.com/#ep01778686b1h0023isopropylalcohol200">
  <j.0:hasCompound rdf:resource=
    "http://rdf.openmolecules.net/?inchi=1/c3h8o/c1-3(2)4/h3-
    4h,1-2h3" />
  <j.0:hasTitle rdf:datatype=
    "http://www.w3.org/2001/XMLSchema#string">
    isopropyl alcohol
  </j.0:hasTitle>
  <j.0:hasAmount rdf:resource=
    "http://www.xml-cml.org/schema#mg_d0660b66763f4dd1" />
  <rdf:type rdf:resource="http://www.patenteye.com/#spectator" />
</rdf:Description>
<rdf:Description rdf:about=
  "http://rdf.openmolecules.net/?inchi=1/c3h8o/c1-3(2)4/h3-4h,1-
  2h3">
  <j.0:hasName rdf:datatype=
    "http://www.w3.org/2001/XMLSchema#string">
    Isopropanol
  </j.0:hasName>
  <j.0:hasInChI rdf:datatype=
    "http://www.w3.org/2001/XMLSchema#string">
    InChI=1/C3H8O/c1-3(2)4/h3-4H,1-2H3
  </j.0:hasInChI>
  <rdfs:subClassOf rdf:resource=
    "http://www.patenteye.com/#ClassName=EPO01015" />
</rdf:Description>
<rdf:Description rdf:about=
  "http://www.patenteye.com/#ClassName=EPO01015">
  <j.0:hasName rdf:datatype=
    "http://www.w3.org/2001/XMLSchema#string">
    low molecular weight aliphatic alcohol
  </j.0:hasName>
</rdf:Description>
<rdf:Description rdf:about=
  "http://www.xml-cml.org/schema#mg_d0660b66763f4dd1">
  <j.0:hasValue rdf:datatype=
    "http://www.w3.org/2001/XMLSchema#float">
    72.20
  </j.0:hasValue>
  <rdf:type rdf:resource="http://www.xml-cml.org/schema#mg" />
</rdf:Description>

```

Figure 6-1: Sample RDF from the PatentEye Repository

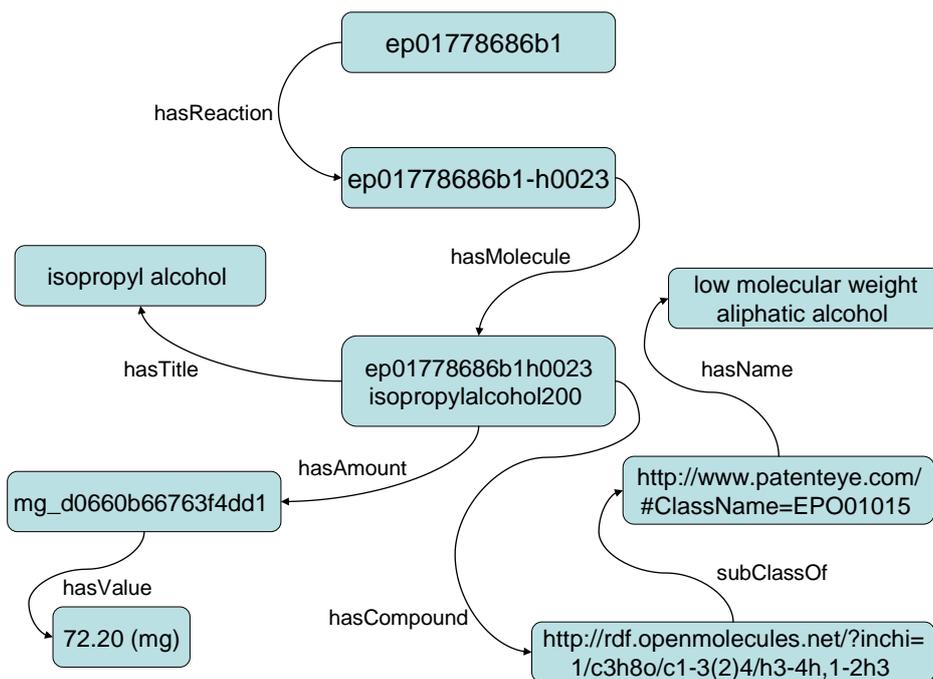


Figure 6-2: Diagrammatic illustration of PatentEye Repository RDF

The examples shown in Figure 6-1 have been heavily trimmed and selected to exemplify the primary features of the PatentEye Repository RDF. This data format will, in the future, allow external parties to access and to interact with the information produced by PatentEye and to incorporate their own data in a highly automated fashion – removing the need for much of the data entry that is currently required when combining datasets. This functionality is predicated upon the usage of the InChI, and the OpenMolecules URIs, to define the chemical species. InChIs lack the ability to represent complex organometallics and polymers (33), for example, and so this technique will not currently be appropriate in all cases but for the majority of small molecules it is likely to be successful.

7. Conclusions

This work has demonstrated how existing and novel technology can be combined to produce machine-understandable chemical information that could be used in support of the semantic web for chemistry and, in particular, how this information can be automatically liberated from existing digital documents. Given the restrictions that surround copyrighted documents, this work was directed towards the patent literature.

Chapter 1 gave an introduction to the problem to be addressed in this thesis, while chapter 2 discussed the existing technologies that were used in the current work and the potential sources of documents that could be used.

Chapter 3 described the extension of Chemical Markup Language to describe Markush structures. Markush structures are of crucial importance to chemical patents, and there currently exists no open standard for describing them. The work in this thesis has demonstrated the creation of semantics, in the form of Extended Polymer Markup Language (EPML), that allow for the description of much of the variability employed in the patent literature, and has demonstrated functioning software for the exemplification and substructure searching of these EPML descriptions. The software and the semantics developed as part of the current work require further refinement before they are published for the community to use, but represent an important proof of concept and initial demonstration of technologies.

Chapter 4 described the implementation of a system for the automatic acquisition of hyponymic relations from the literature. Hyponymic relations form a key part of ontologies – formal representations of knowledge – which play an important role in the Semantic Web. The current manual curation of chemical ontologies such as ChEBI is a time consuming procedure and the creation of a technology that allows semi-automatic curation holds great potential. The system has

been validated by extracting relations from a small patent set, and shown both to operate at a high rate of performance and to discover relations that do not currently form a part of the ChEBI ontology. The technology is consequently recommended for adoption by the community, and current collaboration between the Unilever Centre and the European Bioinformatics Institute is moving in this direction.

Chapters 5 and 6 described the development, implementation and validation of PatentEye – a system for the automatic extraction of chemical reactions from the literature. The extracted data describes the primary products, with attached NMR spectra where possible, and lists the reagents used in the syntheses. The reactions extracted by this method have been manually validated and it is shown that, while the system produces encouraging results, further work is required if there is to be a high level of confidence that the extracted data fully describes the reported reactions. The problem of automatic extraction of chemical reactions is highly non-trivial, and it is unlikely that any system will ever be infallible, but recent developments in the ChemicalTagger tool that assign roles (*e.g.* “add”, “heat”, “wash”) to reaction phrases will enable the assignment of roles to chemical entities detected in an experimental write-up. In turn, this will enable a means to extract not just the components of the reaction but the method too – the recipe as well as the ingredients – in addition to a more reliable means to identify the reagents than at present.

Once an acceptably reliable means of reaction extraction has been implemented, the potential scale on which data can be rapidly extracted from the literature is immense. The current work extracted 4444 reactions from EPO publications covering a period of 10 weeks. With further development increasing the rate of recall from the literature, and the inclusion of patents published by the USPTO and the WIPO, the scale on which PatentEye extracts reactions could easily reach 100,000 reactions/year, while the open access literature offers a further opportunity to increase the scale on which PatentEye operates. Based on the sample of reactions analysed in section 6.1.4, around a third of these might be expected to contain NMR spectra. With the available archive of digitised

patent documents dating back many years, such a collection would immediately dwarf the open NMR database NMRShiftDB (117; 118; 119), which contains around 44000 spectra as of October 2010 and would be of great use to organic chemistry students and researchers as well as to creators of NMR prediction and structure elucidation software.

The reactions derived by this system would provide an excellent overview of the types of chemistry being performed on a small scale in industry. This would allow for the answering of questions that are fascinating on an intellectual and organisational level, such as “how long after a novel method is published is it adopted by industry?” and “what differences exist in the synthetic methods used, by company and by country?” as well as permit the immediate determination of all synthetic routes for a given transformation and the identification of compatible and, by omission to imply incompatible, functional groups in the reactant. For example, the data would be expected to show that ketones may be reduced to alcohols both by LiAlH_4 and NaBH_4 and that their reaction with NaBH_4 is compatible with the presence of an ester group but that the reaction with LiAlH_4 is not. Again, such information is likely to prove useful to organic chemistry students and researchers, and the provision of the data in an open, linked and re-usable form as described in section 6.2 will be of great use to other informatics specialists in the short term and to automated researchers in the longer term.

The software developed during the current work is also likely to prove useful to the community. As discussed in section 5.4, the PatentEye software has already seen use as part of the Green Chain Reaction project. It is hoped that in the future, opportunities will arise to further develop, refine and modularise the software used in the current work so that it may be released under an open licence for the community to reuse freely where appropriate.

8. Bibliography

1. *CAS Database Content at a Glance*. [Online] [Cited: 10 August 2010.] <http://www.cas.org/expertise/cascontent/ataglance/index.html>.
2. *CAS Databases - CAPlus, Journal and Patent References*. [Online] [Cited: 3 November 2010.] <http://www.cas.org/expertise/cascontent/caplus/index.html>.
3. *CAS REGISTRY - The gold standard for substance information*. [Online] [Cited: 3 November 2010.] <http://www.cas.org/expertise/cascontent/registry/index.html>.
4. *CAS Databases - CASREACT, Chemical Reactions*. [Online] [Cited: 3 November 2010.] <http://www.cas.org/expertise/cascontent/casreact.html>.
5. *The Semantic Web*. **T. Berners-Lee, J. Hendler, O. Lassila**. 17 May 2001, Scientific American.
6. *The Automation of Science*. **R. D. King, J. Rowland, S. G. Oliver, M. Young, W. Aubrey, E. Byrne, M. Liakata, M. Markham, P. Pir, L. N. Soldatova, A. Sparkes, K. E. Whelan, A. Clare**. 2009, Science, Vol. 324, pp. 85-89.
7. *Mining chemical structural information from the drug literature*. **D. L. Banville** 2006, Drug Discovery Today, Vol. 11, Issue 1, pp. 35-42.
8. *The Next Big Thing: From Hypermedia to Datuments*. **P. Murray-Rust, H. S. Rzepa**. 2004, Journal of Digital Information, Vol. 5, Issue 1.
9. *Chemistry Add-in for Word*. [Online] [Cited: 4 November 2010.] <http://research.microsoft.com/en-us/projects/chem4word/>.
10. *Computers learn chemistry*. **R. van Noorden**. 2007, *Chemistry World*. Vol. 4, Issue 2, pp. 10.
11. *Project Prospect*. [Online] [Cited: 4 November 2010.] <http://www.rsc.org/Publishing/Journals/ProjectProspect/>.
12. *Semantic enrichment of journal articles using chemical named entity recognition*. **C. R. Batchelor, P. T. Corbett**. 2007. Proceedings of the ACL 2007 Demo and Poster Sessions, Stroudsburg, PA, USA. pp. 45-48.
13. *CrystalEye*. [Online] [Cited: 1 September 2010.] <http://wwmm.ch.cam.ac.uk/crystaleye/>.
14. *ChemicalTagger*. [Online] [Cited: 4 November 2010.] <http://bitbucket.org/lh359/chemicaltagger>.
15. *OSCAR*. [Online] [Cited: 4 November 2010.] <http://sourceforge.net/projects/oscar3-chem/>.
16. *XOM*. [Online] [Cited: 4 November 2010.] <http://www.xom.nu/>.
17. *Chemical Markup Language*. [Online] [Cited: 4 November 2010.] <http://sourceforge.net/projects/cml/>.

18. *CHIC – Converting Hamburgers Into Cows*. **J. A. Townsend, J. Downing, P. Murray-Rust**. 2009. Fifth IEEE International Conference on e-Science, Oxford, UK. pp 337-343.
19. *DSpace@Cambridge*. [Online] [Cited: 12 October 2010.] <http://www.dspace.cam.ac.uk/>.
20. *EThOS - Electronic Theses Online Service*. [Online] [Cited: 12 October 2010.] <http://ethos.bl.uk/>.
21. *USPTO.gov*. [Online] [Cited: 13 August 2010.] http://www.uspto.gov/web/offices/pac/mpep/documents/0600_608_01_v.htm.
22. *Terms and Conditions of Use for the EPO Website*. [Online] [Cited: 13 August 2010.] <http://www.epo.org/etc/termsfuse.html#Copyright>.
23. *EPO - European publication server*. [Online] [Cited: 24 February 2011.] <https://data.epo.org/publication-server>.
24. *EBD Data Information*. [Online] [Cited: 24 February 2011.] <http://docs.epoline.org/ebd/xmlinfo.htm>.
25. *PATENTSCOPE: Search International Patent Applications*. [Online] [Cited: 12 October 2010.] <http://www.wipo.int/pctdb/en/>.
26. *USPTO Bulk Downloads: Patent Grant Full Text*. [Online] [Cited: 13 August 2010.] <http://www.google.com/googlebooks/uspto-patents-grants-text.html>.
27. *XML Path Language (XPath) 2.0*. [Online] [Cited: 6 November 2010.] <http://www.w3.org/TR/xpath20/>.
28. *Open Babel*. [Online] [Cited: 9 August 2010.] http://openbabel.org/wiki/Main_Page.
29. *SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules*. **D. Weininger**. 1988, *J. Chem. Inf. Comput. Sci.*, Vol. 28, pp. 31-36.
30. *SMILES. 2. Algorithm for generation of unique SMILES notation*. **D. Weininger, A. Weininger, J. L. Weininger**. 1989, *J. Chem. Inf. Comput. Sci.*, Vol. 29, pp. 97-101.
31. *InChI Technical Manual*. **S. E. Stein, S. R. Heller, D. V. Tchekhovskoi**. [Online] [Cited: 10 August 2010.] http://www.oci.uzh.ch/edu/lectures/material/DBC/LinNot/InChI_TechMan.pdf.
32. *The IUPAC International Chemical Identifier*. **A. McNaught**. 2006, Chemistry International. Vol. November-December, pp. 12-14.
33. *Unofficial InChI FAQ*. [Online] [Cited: 2 November 2010.] <http://wwmm.ch.cam.ac.uk/inchifaq/#What%20Can%20InChI%20Currently%20Not%20Represent?>.
34. *On a System of Indexing Chemical Literature; Adopted by the Classification Division of the U.S. Patent Office*. **E. A. Hill**, 1900, *J. Am. Chem. Soc.*, Vol. 22, Issue 8, pp. 478-494.
35. *CambridgeSoft Desktop Software - ChemDraw*. [Online] [Cited: 4 November 2010.] <http://www.cambridgesoft.com/software/ChemDraw/>.

36. *JNI-InChI*. [Online] [Cited: 4 November 2010.] <http://jni-inchi.sourceforge.net/>.
37. *Chemical Markup, XML, and the Worldwide Web. 1. Basic Principles*. **P. Murray-Rust, H. S. Rzepa**. 1999, *J. Chem. Inf. Comput. Sci.*, Vol. 39, pp. 928-942.
38. *Chemical Markup, XML, and the World-Wide Web. 2. Information Objects and the CMLDOM*. **P. Murray-Rust, H. S. Rzepa**. 2001, *J. Chem. Inf. Comput. Sci.*, Vol. 41, pp. 1113-1123.
39. *Chemical Markup, XML, and the World-Wide Web. 3. Towards a Signed Semantic Chemical Web of Trust*. **G. V. Gkoutos, P. Murray-Rust, H. S. Rzepa, M. Wright**. 2001, *J. Chem. Inf. Comput. Sci.*, Vol. 41, pp. 1124-1130.
40. *Chemical Markup, XML, and the Worldwide Web. 4. CML Schema*. **P. Murray-Rust, H. S. Rzepa**. 2003, *J. Chem. Inf. Comput. Sci.*, Vol. 43, pp. 757-772.
41. *Chemical Markup, XML, and the World Wide Web. 5. Applications of Chemical Metadata in RSS Aggregators*. **P. Murray-Rust, H. S. Rzepa, M. J. Williamson, E. L. Willighagen**. 2004, *J. Chem. Inf. Comput. Sci.*, Vol. 44, pp. 462-469.
42. *Chemical Markup, XML, and the World Wide Web. 6. CMLReact, an XML Vocabulary for Chemical Reactions*. **G. L. Holliday, P. Murray-Rust, H. S. Rzepa**. 2006, *J. Chem. Inf. Comput. Sci.*, Vol. 46, pp. 145-157.
43. *Chemical Markup, XML, and the World Wide Web. 7. CMLSpect, an XML Vocabulary for Spectral Data*. **S. Kuhn, T. Helmus, R. J. Lancashire, P. Murray-Rust, H. S. Rzepa, C. Steinbeck, E. L. Willighagen**. 2007, *J. Chem. Inf. Comput. Sci.*, Vol. 47, pp. 2015-2034.
44. *CML Sourceforge Repository*. [Online] [Cited: 2 September 2010.] <http://cml.svn.sourceforge.net/viewvc/cml/>.
45. *JUMBOConverters*. [Online] [Cited: 24 February 2011.] <http://bitbucket.org/wwmm/jumbo-converters>.
46. *High-Throughput Identification of Chemistry in Life Science Texts*. **P. Corbett, P. Murray-Rust**. 2006. *Computational Life Sciences II*, Cambridge, UK. pp. 107-118.
47. *ChEBI: a database and ontology for chemical entities of biological interest*. **K. Degtyarenko, P. de Matos, M. Ennis, J. Hastings, M. Zbinden, A. McNaught, R. Alcantara, M. Darsow, M. Guedj, M. Ashburner**. 2008, *Nucleic Acids Research*, Vol. 36, pp. 345-350.
48. *Chemical Entities of Biological Interest: an update*. **P. de Matos, R. Alcantara, A. Dekker, M. Ennis, J. Hastings, K. Haug, I. Spiteri, S. Turner, C. Steinbeck**. 2009, *Nucleic Acids Research*, Vol. 38, pp. 249-254.
49. *Ontology Detail: Physico-chemical methods and properties*. **K. Degtyarenko**. [Online] [Cited: 4 November 2010.] <http://obofoundry.org/cgi-bin/detail.cgi?id=fix>.
50. *Ontology Detail: Physico-chemical process*. **K. Degtyarenko**. [Online] [Cited: 4 November 2010.] <http://www.obofoundry.org/cgi-bin/detail.cgi?id=rex>.

51. *Gene Ontology: tool for the unification of biology*. **The Gene Ontology Consortium**. 2000, *Nature Genet*, Vol. 25, pp. 25-29.
52. *Annotation of Chemical Named Entities*. **P. Corbett, C. Batchelor, S. Teufel**. 2007. *BioNLP: Biological, translational and clinical language processing*, Prague, CZ. pp. 57-64.
53. *Pyridines, pyridine and pyridine rings: disambiguating chemical named entities*. **P. Corbett, C. Batchelor, A. Copestake**. 2008. *LREC Workshop*, Marrakech, MA. pp. 43-50.
54. *Cascaded classifiers for confidence-based chemical named entity recognition*. **P. Corbett, A. Copestake**. 2008, *BMC Bioinformatics*, Vol. 9.
55. *OpenNLP Maxent*. [Online] [Cited: 14 October 2010.] <http://maxent.sourceforge.net/>.
56. *Experimental data checker: better information for organic chemists*. **S. E. Adams, J. M. Goodman, R. J. Kidd, A. D. McNaught, P. Murray-Rust, F. R. Norton, J. A. Townsend, C. A. Waudby**. 2004, *Org. Biomol. Chem.*, Vol. 2, pp. 3067-3070.
57. *Chemical documents: machine understanding and automated information extraction*. **J. A. Townsend, S. E. Adams, C. A. Waudby, V. K. de Souza, J. M. Goodman, P. Murray-Rust**. 2004, *Org. Biomol. Chem.*, Vol. 2, pp. 3294-3300.
58. *ANTLR: ANother Tool for Language Recognition*. [Online] [Cited: 14 October 2010.] <http://www.antlr.org/about.html>.
59. *OpenNLP*. [Online] [Cited: 14 October 2010.] <http://opennlp.sourceforge.net/>.
60. *Building a Large Annotated Corpus of English: The Penn Treebank*. **M. P. Marcus, M. A. Marcinkiewicz, B. Santorini**. 2, 1993, *Computational Linguistics*, Vol. 12, pp. 313-330.
61. *Kekulé: OCR - Optical Chemical (Structure) Recognition*. **J. R. Balmuth, J. R. McDaniel**. 1992, *J. Chem. Inf. Comput. Sci.*, Vol. 32, pp. 373-378.
62. *Chemical Literature Data Extraction: The CLiDE Project*. **P. Ibson, M. Jacquot, F. Kam, A. G. Neville, R. W. Simpson, C. Tonnelier, T. Venczel, A. P. Johnson**. 1992, *J. Chem. Inf. Comput. Sci.*, Vol. 33, pp. 338-344.
63. *Recent Advances in the CLiDE Project: Logical Layout Analysis of Chemical Documents*. **A. Simon, A. P. Johnson**. 1997, *J. Chem. Inf. Comput. Sci.*, Vol. 37, pp. 109-116.
64. *CLiDE Pro: The Latest Generation of CLiDE, a Tool for Optical Chemical Structure Recognition*. **A. T. Valko, A. P. Johnson**. 2009, *J. Chem. Inf. Model.*, Vol. 49, pp. 780-787.
65. *Optical Structure Recognition Software To Recover Chemical Information: OSRA, An Open Source Solution*. **I. V. Filippov, M. C. Nicklaus**. 2009, *J. Chem. Inf. Model.*, Vol. 49, pp. 740-743.
66. *Extracting Chemical Structure Information: Optical Structure Recognition Application*. **I. V. Filippov, M. C. Nicklaus**. 2009. Eighth IAPR International Workshop on Graphics Recognition, La Rochelle, FR. Session 4, pp. 3-12.

67. *Improvements in Optical Structure Recognition Application*. **I. V. Filippov, M. C. Nicklaus, J. Kinney**. 2010. Document Analysis Systems Workshop, Boston, MA, US.
68. *Automated extraction of chemical structure information from digital raster images*. **P. Jungkap, R. Gus, S. Kerby, N. Mandee, L. Naesung, S. Kazuhiro**. 2009, Chemistry Central Journal, Vol. 3.
69. *Towards in-house searching of Markush structures from patents*. **J. M. Barnard, P. M. Wright**. 2009, World Patent Information, Vol. 31, pp. 97-103.
70. *The Chemical Abstracts Service Generic Chemical (Markush) Structure Storage and Retrieval Capability. 1. Basic Concepts*. **W. Fisanick**. 1990, J. Chem. Inf. Comput. Sci., Vol. 30, pp. 145-154.
71. *The Chemical Abstracts Service Generic Chemical (Markush) Structure Storage and Retrieval Capability. 2. The MARPAT File*. **T. Ebe, K. A. Sanderson, P. S. Wilson**. 1991, J. Chem. Inf. Comput. Sci., Vol. 31, pp. 31-36.
72. *Computer Storage and Retrieval of Generic Structures in Chemical Patents. 1. Introduction and General Strategy*. **M. F. Lynch, J. M. Barnard, S. M. Welford**. 1981, J. Chem. Inf. Comput. Sci., Vol. 21, pp. 148-150.
73. *Computer Storage and Retrieval of Generic Structures in Chemical Patents. 2. GENSAL, a Formal Language for the Description of Generic Chemical Structures*. **J. M. Barnard, M. F. Lynch, S. M. Welford**. 1981, J. Chem. Inf. Comput. Sci., Vol. 21, pp. 151-161.
74. *Computer representation and manipulation of combinatorial libraries*. **J. M. Barnard, G. M. Downs**. 1997, Perspectives in Drug Discovery and Design, Vol. 7/8, pp. 13-30.
75. *Chemical Markup, XML and the World-Wide Web. 8. Polymer Markup Language*. **N. Adams, J. Winter, P. Murray-Rust, H. S. Rzepa**. 2008, J. Chem. Inf. Model., Vol. 48, pp. 2118-2128.
76. *Polymer Builder*. [Online] [Cited: 28 October 2010.] <http://wwmm-svc.ch.cam.ac.uk/polydemo/>.
77. *Properties of Polymers: Their Correlation with Chemical Structure; their Numerical Estimation and Prediction from Additive Group Contributions*. **D. W. van Krevelen**. 1997. 3rd Revised edition, Elsevier Science Ltd.
78. *The Number of Structurally Isomeric Alcohols of the Methanol Series*. **H. R. Henze, C. M. Blair**. 1931, J. Am. Chem. Soc., Vol. 53, Issue 8, pp. 3042-3046.
79. *A Comparison of Different Approaches to Markush Structure Handling*. **J. M. Barnard**. 1991, J. Chem. Inf. Comput. Sci., Vol. 31, pp. 64-68.
80. *Computer Storage and Retrieval of Generic Chemical Structures in Patents. 3. Chemical Grammars and Their Role in the Manipulation of Chemical Structures*. **S. M. Welford, M. F. Lynch, J. M. Barnard**. 1981, J. Chem. Inf. Comput. Sci., Vol. 21, pp. 161-168.
81. *CORINA - Fast Generation of High-Quality 3D Molecular Models*. [Online] [Cited: 4 November 2010.] <http://www.molecular-networks.com/products/corina>.

82. *Recent Developments of the Chemistry Development Kit (CDK) - An Open-Source Java Library for Chemo- and Bioinformatics*. **C. Steinbeck, C. Hoppe, S. Kuhn, M. Floris, R. Guha, E. L. Willighagen**. 2006, *Curr. Pharm. Des.*, Vol. 12, Issue 17, pp. 2111-2120.
83. *The Chemistry Development Kit (CDK): An Open-Source Java Library for Chemo- and Bioinformatics*. **C. Steinbeck, Y. Han, S. Kuhn, O. Horlacher, E. Luttmann, E. Willighagen**. 2003, *J. Chem. Inf. Comput. Sci.*, Vol. 43, pp. 493-500.
84. *Sourceforge.net: CDK*. [Online] [Cited: 4 November 2010.] http://sourceforge.net/apps/mediawiki/cdk/index.php?title=Main_Page.
85. *Jmol: an open-source Java viewer for chemical structures in 3D*. [Online] [Cited: 4 November 2010.] <http://www.jmol.org/>.
86. *Computer Storage and Retrieval of Generic Structures in Chemical Patents. 4. An Extended Connection Table Representation for Generic Structures*. **J. M. Barnard, M. F. Lynch, S. M. Welford**. 1982, *J. Chem. Inf. Comput. Sci.*, Vol. 22, pp. 16-164.
87. *A Relaxation Algorithm for Generic Chemical Structure Screening*. **A. von Scholley**. 1983, *J. Chem. Inf. Comput. Sci.*, Vol. 24, pp. 245-241.
88. *Computer Storage and Retrieval of Generic Structures in Chemical Patents. 5. Algorithmic Generation of Fragment Descriptors for Generic Structure Screening*. **S. M. Welford, M. F. Lynch, J. M. Barnard**. 1983, *J. Chem. Inf. Comput. Sci.*, Vol. 24, pp. 57-66.
89. *Computer Storage and Retrieval of Generic Structures in Chemical Patents. 8. Reduced Chemical Graphs and Their Application in Generic Chemical Structure Retrieval*. **V. J. Gillet, G. M. Downs, A. Ling, M. F. Lynch, P. Venkataram, J. V. Wood**. 1987, *J. Chem. Inf. Comput. Sci.*, Vol. 27, pp. 126-137.
90. *Marvin: History of Changes*. [Online] [Cited: 8 October 2010.] <http://www.chemaxon.com/marvin/help/developer/changes.html>.
91. *Automatic Acquisition of Hyponyms from Large Text Corpora*. **M. A. Hearst**. 1992. 14th Conference on Computational Linguistics, Nantes, FR. pp 539-545.
92. *Facts from text: can text mining help to scale-up high-quality manual curation of gene products with ontologies?* **R. Winnenburger, T. Wachter, C. Pike, A. Doms, M. Schroeder**. 2008, *Briefings in Bioinformatics*, Vol. 9, Issue 6, pp. 466-478.
93. *OWL Web Ontology Language Overview*. [Online] [Cited: 1 November 2010.] <http://www.w3.org/TR/owl-features/>.
94. *OWL API*. [Online] [Cited: 24 February 2011.] <http://owlapi.sourceforge.net/>.
95. *ChEBI FAQ*. [Online] [Cited: 13 November 2010.] <http://www.ebi.ac.uk/chebi/faqForward.do#2>.
96. *EPO - Basic definitions*. [Online] [Cited: 24 February 2011.] <http://www.epo.org/patents/patent-information/european-patent-documents/basic-definitions.html>.

97. *European Publication Server Data Coverage*. [Online] [Cited: 1 November 2010.] <https://data.epo.org/publication-server/data-coverage>.
98. *Guide for Applicants*. [Online] [Cited: 24 February 2011.] <http://www.epo.org/patents/Grant-procedure/Filing-an-application/European-applications/Guide-for-applicants.html>.
99. *Download OSCAR3 from SourceForge.net*. [Online] [Cited: 5 November 2010.] <http://sourceforge.net/projects/oscar3-chem/files/oscar3-chem/alpha5/oscar3-a5.zip/download>.
100. *Chemical documents: machine understanding and automated information extraction*. **J. A. Townsend, S. E. Adams, C. A. Waudby, V. K. de Souza, J. M. Goodman, P. Murray-Rust**. 2004, *Org. Biomol. Chem.*, Vol. 2, pp. 3294-3300.
101. *Classifier4J*. [Online] [Cited: 24 February 2011.] <http://classifier4j.sourceforge.net/>.
102. *ImageMagick: Convert, Edit and Compose Images*. [Online] [Cited: 5 November 2010.] <http://www.imagemagick.org/>.
103. *Optical Structure Recognition Software To Recover Chemical Information: OSRA, An Open Source Solution*. **I. V. Filippov, M. C. Nicklaus**. 2009, *J. Chem. Inf. Model.*, Vol. 49, pp. 740-743.
104. *Extraction of Chemical Reaction Information from Primary Journal Text Using Computational Linguistics Techniques. 1. Lexical and Syntactic Phases*. **E. M. Zamora, P. E. Blower Jr**. 1984. *J. Chem. Inf. Comput. Sci.*, Vol. 24, pp. 176-181.
105. *Extraction of Chemical Reaction Information from Primary Journal Text Using Computational Linguistics Techniques. 2. Semantic Phase*. **E. M. Zamora, P. E. Blower Jr**. 1984. *J. Chem. Inf. Comput. Sci.*, Vol. 24, pp. 181-188.
106. *Extraction of Chemical Reaction Information from Primary Journal Text*. **C. S. Ai, P. E. Blower Jr, R. H. Ledwith**. 1990. *J. Chem. Inf. Comput. Sci.*, Vol. 30, pp. 163-169.
107. *IUPAC Compendium of Chemical Terminology (the "Gold Book")* **A. D. Wilkinson, A. McNaught**. 1997. 2nd edition, Blackwell Scientific Publications.
108. *chemicx.com* [Online] [Cited: 5 November 2010.] <http://www.chemicx.com/>.
109. *Green Chain Reaction - Science Online London 2010*. [Online] [Cited: 27 October 2010.] <http://scienceonlinelondon.wikidot.com/topics:green-chain-reaction>.
110. *Chemical Name to Structure: OPSIN, an Open Source Solution*. **D. M. Lowe, P. T. Corbett, P. Murray-Rust, R. C. Glen**. *J. Chem. Inf. Model.* Manuscript accepted.
111. *Amide Substituted Xanthine Derivatives With Gluconeogenesis Modulating Activity*. **EP 1515972. P. W. Dunten, L. H. Foley, N. J. S. Huby, S. L. Pietranico-Cole**. 2003.
112. *RDF/XML Syntax Specification (Revised)*. [Online] [Cited: 1 November 2010.] <http://www.w3.org/TR/REC-rdf-syntax/>.
113. *OpenRDF.org*. [Online] [Cited: 1 November 2010.] <http://www.openrdf.org/>.

114. *SPARQL Query Language for RDF*. [Online] [Cited: 1 November 2010.] <http://www.w3.org/TR/rdf-sparql-query/>.
115. *PatentEye Repository*. [Online] [Cited: 29 October 2010.] <http://bitbucket.org/lh359/patenteye>.
116. *OpenMolecules.net*. [Online] [Cited: 1 November 2010.] <http://openmolecules.net/>.
117. *NMRShiftDB - open nmr on the web*. [Online] [Cited: 1 November 2010.] <http://www.ebi.ac.uk/nmrshiftdb>.
118. *NMRShiftDB - Constructing a Free Chemical Information System with Open-Source Components*. **C. Steinbeck, S. Krause, S. Kuhn**. 2003, *J. Chem. Inf. Comput. Sci.*, Vol. 43, pp. 1733-1739.
119. *NMRShiftDB - compound identification and structure elucidation support through a free community-built web database*. **C. Steinbeck, S. Kuhn**. 2004, *Phytochemistry*, Vol. 65, pp. 2711-2717.

Appendix A

Hyponymic Relations

Hyponymic (“is-a”) relations exist between two terms where one, the hyponym, is a subset of the other, the hypernym. For example, “vehicle” is a hypernym of “car”, and “Ford Fiesta” is a hyponym of “car”. Hypernyms and hyponyms may take the form of single words or of phrases, as shall be seen later. This task aims to quantify the performance of a system for the automatic acquisition of such relations based on *Hearst Patterns*.

Hearst Patterns

Hearst first proposed the use of lexico-syntactic patterns for the automatic acquisition of hyponymic relations, thereafter known as *Hearst Patterns*. She described six patterns that could be employed;

Format	Example	Pattern Name
HYPER such as <i>HYPO</i>	Apolar solvents such as <i>THF</i> and <i>hexane</i>	SUCH_AS
such HYPER as <i>HYPO</i>	Such bases as <i>NaOEt</i> or <i>LDA</i>	SUCH_FOO_AS
<i>HYPO</i> or other HYPER	<i>MeCl</i> , <i>EtBr</i> or other organohalides	OR_OTHER
<i>HYPO</i> and other HYPER	<i>Benzene</i> , <i>ethylene oxide</i> and other carcinogens	AND_OTHER
HYPER including <i>HYPO</i>	Methyl ketones including <i>acetone</i>	INCLUDING
HYPER especially <i>HYPO</i>	Grignard reagents , especially <i>methyl magnesium chloride</i>	ESPECIALLY

wherein **HYPER** indicates the hypernym, indicated in bold and *HYPO* the hyponym(s), indicated in italics.

Each of these patterns has been assigned a name for ease of reference. In the example for the SUCH_AS pattern, the text communicates the information that THF and hexane are examples of apolar solvents. This information is readily available to a fluent speaker of the English language, regardless of whether or not they are aware of what “THF”, “hexane” or an “apolar solvent” are.

The Task

- This task focuses solely on the SUCH_AS pattern. You will be presented with a series of paragraphs, each containing one or more instances of the phrase “such as”. You should read each paragraph and identify the Hearst Pattern(s). For each Hearst Pattern you should identify the hypernym and the hyponym(s). You should use your own judgement to determine where in the text the pattern and the hypernym and hyponym(s) begin and end, based on the guidelines that follow and using your background knowledge or other sources to ensure that the hyponymic relations you find are correct. Thus, in the case of “esters of

unsaturated carboxylic acids such as *maleic acid*”, the hypernym is “unsaturated carboxylic acids”, not “esters of unsaturated carboxylic acids”.

- A Hearst Pattern must be composed of a single hypernym, the phrase “such as”, a single hyponym or a list of hyponyms, optionally including leading determiners (e.g. “a”, “the”, “some”, “any”), and nothing else. You should not annotate a leading determiner as part of a hyponym unless you consider it vital to the meaning of the hyponym term. Thus, for example, you should annotate “**stable carbocations** such as the *tertiary carbocation*”.
- Where more than one hyponym is found the list must be continuous. Whitespace, punctuation and conjunctions (“and” and “or”) are allowed to separate the list, other words are not. Thus, “polar solvents such as, for example, DMSO” should not be annotated. Where this extra text occurs inside a list of hyponyms, e.g. “**polar solvents** such as *DMSO*, *THF* – the most commonly used solvent – and acetone”, you should annotate the hyponyms that occur prior to the extra text and the Hearst Pattern as far as the end of the last annotated hyponym.
- We are looking specifically for Hearst Patterns that inform us about the chemical domain. You should only include patterns in which all of the hyponyms are terms that have structural meaning, such as chemical structures, structural features or structural classes. Hyponyms thus need not correspond to specific chemicals, so for example may include “methyl group” and “beta-lactams” as well as specific molecules. Specific molecules may be identified by, for example, trivial names, systematic names, semi-systematic names, abbreviations such as “DMAP” or formulae such as “C₃H₈” or “MeCOCH₂Cl”.
- Hypernyms should therefore denote classes of chemical structures, structural features or structural classes. Hypernyms may themselves be structural classes (e.g. “beta-lactams”), but may also be based on function (e.g. “5-HT antagonists”), usage (e.g. “solvents”), properties (e.g. “visible-light absorbing molecules”) or something far more ephemeral (e.g. “interesting functional groups”). These examples should be considered illustrative rather than restrictive.
- Hypernyms may include adjectives where the adjective forms a part of the hypernym, e.g. in “...using a **non-polar solvent** such as *cyclohexane*”, but not in “...by dissolution in cyclohexane or an alternative **solvent** such as *benzene*”.
- Where suitable patterns are found, annotate the entire text of the pattern by using “click-and-drag” to select the appropriate text within the OSCAR scrapbook and clicking the “pattern” button. Then annotate the hypernym and the individual hyponyms similarly by using the “hyper” and “hypo” buttons respectively.
- All annotations must begin and end at word boundaries.
- It is not necessarily the case that all Hearst Patterns may be annotated in this way. Consider, for example, “metal oxides such as potassium and calcium oxide”. “Potassium and calcium oxide” describes the hyponyms, but potassium is not a metal oxide, and “potassium oxide” cannot be annotated as the words do not occur together. In this case you should annotate “metal oxide” as the hypernym, and “calcium oxide” as the **only** hyponym. This point should be applied where the final word significantly modifies the meaning of the non-final items in the list and not to cases such as “**alkoxide anions** such as *methoxide* and *ethoxide* anions”.
- If the phrasing used in the text makes understanding the meaning of a Hearst Pattern impossible, you should not annotate anything.

- Hearst patterns may use a hypernym denoting multiple classes, *e.g.* “**polar or non-polar solvents** such as *DMSO* or *hexane*”. In this case, annotate the hypernym as “polar or non-polar solvents” and the hyponyms as normal.
- If there are nested Hearst Patterns, *e.g.* “...antibiotics such as beta-lactams such as amoxicillin...” then copy the source file as many times as necessary and annotate the patterns separately.
- If there is a typo present in a hypernym or hyponym, you should treat the word as though the typo were not there. If there is a typo in the phrase “such as” you should not annotate the Hearst Pattern.

Appendix B

The table on the following page contains a list of classes of molecules derived from the application of Hearst Patterns to chemical texts. In this task, each class should be assigned one and only one of the following labels;

- Structural – the name of the class indicates that all members contain a specific substructure *e.g.* ketone or methyl ester.
- Functional – the name of the class indicates that all members share a common function, usage, property or other non-structural feature *e.g.* antibiotic or surfactant.
- Semi-structural – the name of the class indicates something about the structure or composition of the members, but not that they share a specific substructure *e.g.* isomers of C₆H₁₀O or bicyclic systems.

Having decided which of the labels fits the class name best, tick the appropriate box. In making your decision you may use your background knowledge and any reference sources you consider appropriate but you **must not** confer with anyone.

Class name	Structural	Functional	Semi-structural
olefin			
straight monoolefin			
amine			
inert solvent			
ether compound			
mineral and carboxylic acid			
alkylamine			
suitable solvent			
trihydrocarbon-substituted phosphine			
aromatic ether compound			
halogenated styrene			
hydrocarbon			
organic solvent			
halogenated α -olefin			
organic acid			
monohydrocarbon-substituted phosphine			
diolefin			
Base			
α -olefin			
solvent			
alkylstyrene			
alcohol			
dihydrocarbon-substituted phosphine			
cyclic olefin			
tertiary amine			
halogenated hydrocarbon			
aliphatic monoether compound			
aliphatic unsaturated ether compound			

Appendix C

1. Annotate textual reports of spectral data. Do not annotate spectra that are reported in image form. Include spectra found within tables where it is possible to produce well-formed XML and where they should be annotated according to these guidelines.
2. Spectra should be annotated such that each `spectrum` tag contains one and only one spectrum.
3. Annotate spectra that are reported in a regular format, *e.g.* “¹H NMR: 2.30 (s, 2H), 2.45 (d, 1H, J=2.8Hz)”. Do not include spectra reported in natural language *e.g.* “¹H NMR found to be identical as for previous example”.
4. Spectra containing typos should be annotated as though the typo were not present.
5. Do not include leading or trailing whitespace or punctuation.
6. Annotate only text corresponding to spectra produced from ¹H NMR (HNMR), ¹³C NMR (CNMR), Mass Spectrometry (MassSpec), High-Resolution Mass Spectrometry (HRMS) and Infra-Red Spectroscopy (IR).

Appendix D

Supporting information is contained on the attached disk. This information comprises;

- `/code/markush` - the implementation of EPML tools as described in chapter 3
 - `/code/markush/examples` - example EPML describing different types of variation
 - `/code/markush/fragments` - CML fragments describing assorted key units
 - `/code/markush/markushStructures` - example markush structures encoded in EPML
 - `/code/markush/polyinfo` - PML descriptions of polymers from the PoLyInfo database and their associate CML fragments and atomistic CML descriptions
 - `/code/markush/src` - the source code for the implementation
 - `/code/markush/test` - test code that really should have been in `src/test/java`
 - `/code/markush/testResources` - test files that really should have been in `src/test/resources`
-
- `/code/patentanalysis` - the implementations of Hearst Patterns for relation extraction, and of reaction extraction described in chapters 4 and 5&6 respectively
 - `/code/patentanalysis/classifier` - the experimental sections, sorted according to their status as "experimental", "non-experimental" and "empty" used for model training and validation, as described in section 5.2.4
 - `/code/patentanalysis/downloadsNoDuplicates` - the ten weeks' worth of EPO patent downloads from which duplicated documents have been removed, as described in section 5.1.3
 - `/code/patentanalysis/osra` - the OSRA executable and supporting libraries used for image recognition
 - `/code/patentanalysis/osraCache` - the cache of OSRA results generated during the work
 - `/code/patentanalysis/src` - the source code for the implementation
 - `/code/patentanalysis/testResources` - test files that really should have been in `src/test/resources`
-
- `/data` - data sets produced during the work
 - `/data/chemImageCorpus` - the corpus of chemical images used to measure the accuracy of OSRA in section 5.2.5, produced by random selection from the files in `/data/processedPatents`
 - `/data/dataAnnotations` - the corpus of experimental paragraphs annotated for experimental data as described in section 5.2.3.2, produced by random selection from the files in `/data/processedPatents`
 - `/data/finalReactions` - the resulting CML reactions from reaction mining the patent corpus, and the reasons for failure as discussed in section 5.3.3, by executing ReactionExtractor on the files in `/data/processedPatents`
 - `/data/hearstAnnotations` - the sets of annotations produced by the three annotators as discussed in section 4.3.5
 - `/data/processedPatents` - the set of semantically enhanced patents produced from the set of unique, full text documents as described in section 5.2, produced by executing ProcessPatents on the files in `/code/patentanalysis/downloadsNoDuplicates`
 - `/data/reactionCorpus` - the set of automatically extracted reactions used to validate the performance of PatentEye in section 6.1 (produced by random selection from the files in

/data/finalReactions), and an index file containing the manually-determined tp, fp and fn scores per patent

- /data/generatedRelations - the OWL files containing the sets of relations used in chapter 4
- /data/generatedRelations/structureOntology.owl - the full set of relations extracted from the patents, generated by running StructureOntologyCreator on the patents contained in /code/patentanalysis/downloadsNoDuplicates as described in section 4.3
- /data/generatedRelations/trimmedBySourceCount.owl - the set of relations produced by trimming the relations to those asserted in 3 separate source documents, as described in section 4.3.4, by executing TrimStructureOntology on structureOntology.owl
- /data/generatedRelations/trimmedBySubclasses.owl - the set of relations produced by further trimming the relations to only include those that refer to classes with at least six example structures, as described in section 4.4.1, by executing TrimStructureOntology on structureOntology.owl

Appendix E

The following comprises a list of the major software components written solely by the current author as part of the current work. Much of the code is available on the attached disk.

- The `MarkushBuilder` component for producing example structures corresponding to Markush structures encoded in EPML, as described in section 3.4.
- The software for compiling and searching Extended Connection Tables corresponding to Markush structures encoded in EPML, as described in section 3.5.
- The `HearstFinder` component for identifying chemical Hearst Patterns in text using `ChemicalTagger`, as described in section 4.3.1
- The `EPOCrawler` component for automatically downloading chemical patents from the European Patent Office (EPO) website, as described in section 5.1.2.
- The `EPOProcessor` component for the semantic enhancement of EPO patent documents, as described in section 5.2.
- The `ExperimentParser` component for the automatic extraction and semantic description of chemical syntheses from plain text, as described in section 5.3.