DEPARTMENT OF CHEMISTRY

Extraction of chemical structures and reactions from the literature

Daniel Mark Lowe

Pembroke College

This dissertation is submitted for the degree of Doctor of Philosophy

June 2012

Disclaimer

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 Chemistry Degree Committee.

Abstract

The ever increasing quantity of chemical literature necessitates the creation of automated techniques for extracting relevant information. This work focuses on two aspects: the conversion of chemical names to computer readable structure representations and the extraction of chemical reactions from text.

Chemical names are a common way of communicating chemical structure information. OPSIN (Open Parser for Systematic IUPAC Nomenclature), an open source, freely available algorithm for converting chemical names to structures was developed. OPSIN employs a regular grammar to direct tokenisation and parsing leading to the generation of an XML parse tree. Nomenclature operations are applied successively to the tree with many requiring the manipulation of an in-memory connection table representation of the structure under construction. Areas of nomenclature supported are described with attention being drawn to difficulties that may be encountered in name to structure conversion. Results on sets of generated names and names extracted from patents are presented. On generated names, recall of between 96.2% and 99.0% was achieved with a lower bound of 97.9% on precision with all results either being comparable or superior to the tested commercial solutions. On the patent names OPSIN's recall was 2-10% higher than the tested solutions when the patent names were processed as found in the patents. The uses of OPSIN as a web service and as a tool for identifying chemical names in text are shown to demonstrate the direct utility of this algorithm.

A software system for extracting chemical reactions from the text of chemical patents was developed. The system relies on the output of ChemicalTagger, a tool for tagging words and identifying phrases of importance in experimental chemistry text. Improvements to this tool required to facilitate this task are documented. The structure of chemical entities are where possible determined using OPSIN in conjunction with a dictionary of name to structure relationships. Extracted reactions are atom mapped to confirm that they are chemically consistent. 424,621 atom mapped reactions were extracted from 65,034 organic chemistry USPTO patents. On a sample of 100 of these extracted reactions chemical entities were identified with 96.4% recall and 88.9% precision. Quantities could be associated with reagents in 98.8% of cases and 64.9% of cases for products whilst the correct role was assigned to chemical entities in 91.8% of cases. Qualitatively the system captured the essence of the reaction in 95% of cases. This system is expected to be useful in the creation of searchable databases of reactions from chemical patents and in facilitating analysis of the properties of large populations of reactions.

Acknowledgements

I would like to thank my supervisors, Professor Robert Glen and Professor Peter Murray-Rust, for their guidance and advice. I would also like to thank Dr Peter Corbett, for his initial work on the OPSIN codebase which was the precursor to the system that I developed, Dr Lezan Hawizy for her work on ChemicalTagger and many useful discussions on extending it, Dr David Jessop for his paragraph classifier, Albina Asadulina for her contribution to fused ring nomenclature support and Dr Sam Adams for many fruitful discussions on cheminformatics algorithms. I would also like to thank my colleagues at the Unilever Centre for providing such an enjoyable working environment. I am very grateful to Boehringer Ingelheim for funding my research.

Table of Contents

Glossary

AST = Abstract Syntax Tree

CAS = Chemical Abstracts Service

ChEBI = Chemical Entities of Biological Interest

CIP = Cahn-Ingold-Prelog

CML = Chemical Markup Language

DTD = Document Type Definition

EPO = European Patent Office

InChI = IUPAC International Chemical Identifier

IPC = International Patent Classification

IUPAC = International Union of Pure and Applied Chemistry

MEMM = Maximum-Entropy Markov Model

OCR = Optical Character Recognition

OPSIN = Open Parser for Systematic IUPAC Nomenclature

OSCAR = Open Source Chemistry Analysis Routines

POS = Part Of Speech

SMILES = Simplified Molecular Input Line Entry Specification

USPTO = United States Patent and Trademark Office

WIPO = World Intellectual Property Organization

XML = eXtensible Markup Language

Chapter 1 Introduction

The scientific literature, comprising journal articles (Figure 1-1), patents (Figure 1-2) and theses, is continuing to grow rapidly.

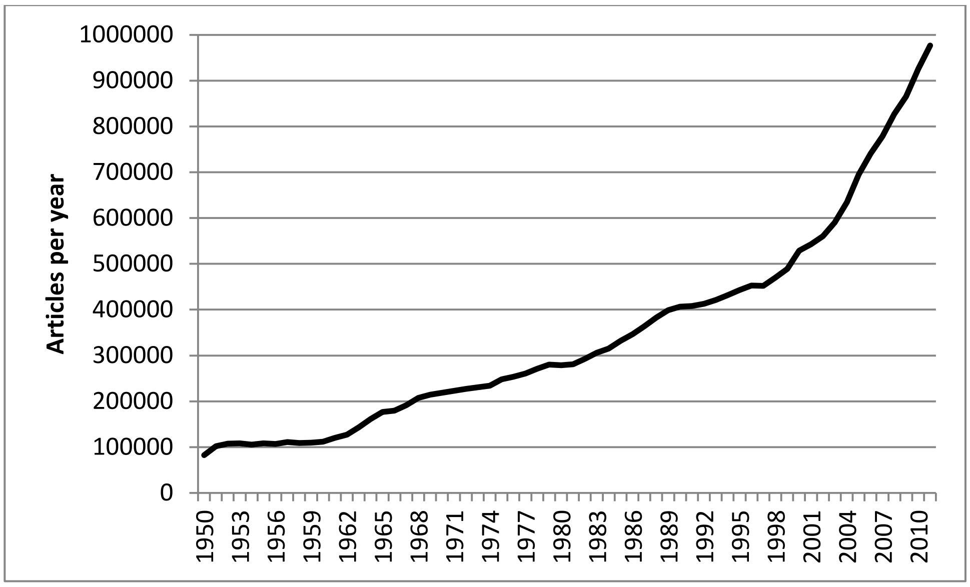


Figure 1-1 PubMed articles indexed per year from 1950-2011[1]

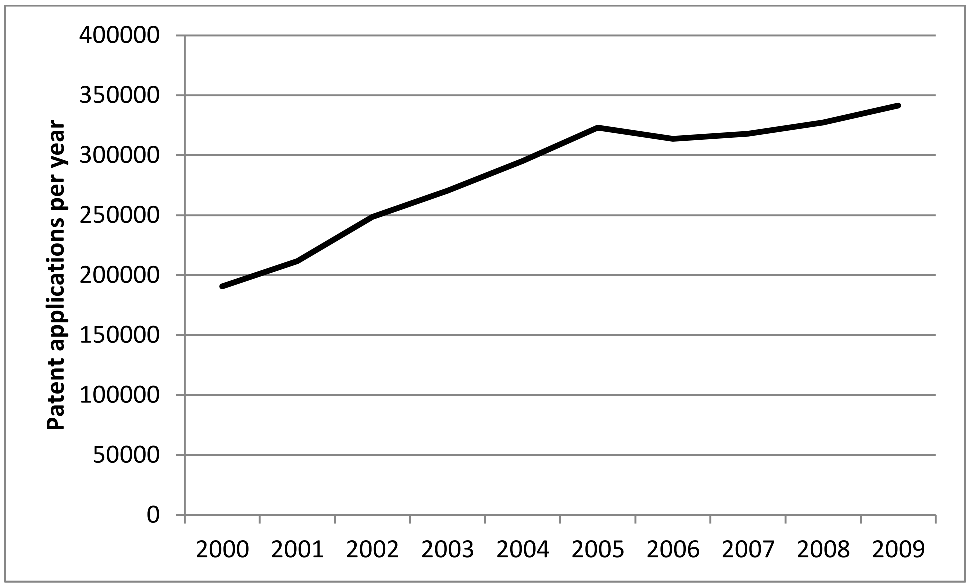


Figure 1-2 World-wide chemistry patent applications per year from 2000-2009[2]

Due to the size of the literature automated methods must be employed to allow identification of relevant resources. Fortunately much of the literature is available in digital form whether by being natively created as such or, in the case of legacy material, by being scanned. Optical character recognition (OCR) is routinely employed on scanned documents to allow their text to be computer readable. Using modern search engines, employing technologies such as Apache Lucene[3], full text searching is now routine; however such searches are not sufficient to allow domain specific queries as traditional search engines have limited understanding of the content of the documents over which they are searching.

Ontologies may be employed to formally encode the relation between entities in a domain e.g. those that are hyponyms of other entities, and to encode terms that are synonymous with a given concept in the ontology. Examples of such ontologies include the Gene Ontology[4] and the ChEBI[5] (Chemical Entities of Biological Interest) ontology. Unlike some fields, the number of possible entities in chemistry is essentially unbounded. For example, the number is of the order of 10^{60}–10^{100} just for drug-like small molecule entities[6]. This, coupled with the use of various forms of systematic nomenclature that lead to many names for the same chemical entity, makes the existence of an ontology describing all possible chemical entities and their possible synonyms impractical.

For small molecules, a natural identifier is the chemical structure itself and hence much text mining effort in chemistry has focused on the identification and conversion of textual and graphical entities into chemical structures.

Textual chemical entities may be expressed in many ways including systematic nomenclature such as IUPAC nomenclature, trivial names, chemical line identifiers e.g. InChI (Section 2.4) and chemical formulae. In biomedical text mining identification of entities can be primarily achieved by dictionary-based approaches[7]. In chemical text mining, however, dictionary-based approaches are insufficient to recognise much systematic nomenclature, line identifiers and chemical formulae. As a result, recent research on the identification of chemical entities from text has focused primarily on machine-learning approaches[8–15]. More recently, grammar-based approaches have also been shown to be applicable[16].

Just as for the identification of textual chemical entities, resolution to chemical structures, in the general case cannot be accomplished by dictionary approaches. As a result chemical name to structure algorithms are required to allow the interpretation of systematic chemical nomenclature.

Corresponding efforts exist to extract chemical structures from images. Eight different solutions have been reported, two of which are currently open source, that are under active development [17–24]. The area is rapidly progressing with new versions and new solutions significantly increasing the percentage of chemical structure diagrams that can be recognised correctly. For example, using one particular test set 69%[25] (OSRA, 2009) to 88%[22] (MolRec, 2012). With the growing maturity of image to structure software, future research can be expected to increasingly leverage the combination of the results of text mining and image to structure[26].

Due to concerns over the accuracy of image to structure software, at the time the project was initiated, and the pre-existence of an actively developed open source image to structure solution it was decided to focus this research project solely on extracting information from text.

The lack of an open source name to structure algorithm with useful levels of performance necessitated the development of such a name to structure algorithm as a critical part of this project. This forms the first part of this project.

1.1 Where can text mining be performed?

Much of the chemical literature published in journals remains behind pay-walls with policies on text mining that differ significantly between publishers and, often with restrictions and/or charges attached[27]. A further problem is that no standardised data format exists for the representation of journal articles meaning that some level of adaption is likely to be required for a tool to work with articles from a particular journal.

Patents also provide a vast resource of chemical information, yet have the key advantage of being in the public domain. More recent patents also have the advantage of being presented in standardised data formats. Historically, access to bulk patent downloads has been difficult, but, with the recent collaboration between the USPTO (United States Patent and Trademark Office) and Google Patents[28], the entire archive of US Patents can now be downloaded trivially.

Other important sources of patents include the European Patent Office (EPO), Japan Patent Office, Korean Intellectual Property Office and the State Intellectual Property Office of the People's Republic of China. It should, however, be considered that the most important patents are likely to be filed at multiple patent offices. The World Intellectual Property Organization (WIPO) is an important source of patent applications that are intended to be processed at multiple patent offices but have not yet reached the “national phase” where they are examined by the national patent offices.

Due to their ease of access and lack of OCR mistakes, USPTO patent applications from 2008 to 2011 forms the corpus used for text mining in this project.

1.2 What can be text mined?

Text mining has seen widespread use in bioinformatics for discovering relationships between entities e.g. chemical interactions with Cytochrome P450 [29–31] or protein-protein interactions[32]. Uses in chemistry have been more limited including annotation of entities[33] (*cf*. the RSC's Project Prospect), the association of linked and/or calculated data with identified entities[34,35] (*cf*. ChemAxon's Chemicalize), and allowing patents to be structure searchable though large scale extraction of chemical structures (*cf*. SureChem[36], IBM BAO strategic IP insight platform[37]).

No large scale attempt at automatically extracting reactions from the literature has been attempted which is the problem this project will address. Such a system has the potential to allow more precise queries of the mined reactions and to improve knowledge driven reaction prediction algorithms[38].

1.3 Overview of research project

Chapter 2 describes the theory and software solutions that underlie the solutions that have been developed. Covered topics include, computer readable chemical structure serialisations, grammars and automata, software developed for identifying chemical entities and annotating experimental chemistry text, and techniques that help provide a productive software development environment.

Chapter 3 describes the development of OPSIN, a chemical name to structure algorithm. Other existing and historic attempts at name to structure are discussed followed by a detailed description of the processes that allow OPSIN to convert a name into a computer readable structure representation. The forms of nomenclature supported by OPSIN are described, exemplified and, where of sufficient complexity, the algorithms used to process them are described. OPSIN's performance is evaluated on sets of generated chemical names and names extracted from patents. The various ways that OPSIN is used including as a command-line interface, web service and tool for identifying systematic chemical names in free text are described.

Chapter 4 describes the development of software for the automated extraction of reactions from patents. Previous attempts are discussed followed by a detailed description of the reaction extraction system that has been developed. This covers the steps of identifying experimental sections, determining the type and role of chemical entities and finally producing an atom-atom map between the reactants and product/s. Precision and recall estimates are derived from a subset of the four years of USPTO patents over which the system has been run.

Chapter 5 summarises the outcomes and future directions of this project.

Chapter 2 Tools and Methods

2.1 XML

XML (eXtensible Markup Language) is a standard for encoding information using mark-up in a way that is machine-readable. An XML document may contain *element*s, *attribute*s, *text* nodes, *comment*s, *processing instruction*s, *namespace* declarations and *doctype* declarations. To explain the first three of these, Figure 2-1 will be used as an example. The document is formed of *elements*, made of labelled start and end tags, in this case `inventory` and `vehicle`. An element may be associated with zero or more *attributes* e.g. `wheels`. An element may also have zero or more *text* children e.g. 'car'. To be well formed i.e. valid, an XML document must have a single root from which all other *elements* are ultimately descendants.

```
<inventory>
 <vehicle wheels="4">car</vehicle>
 <vehicle wheels="3">tricycle</vehicle>
 <vehicle wheels="2">bicycle</vehicle>
</inventory>
```

Figure 2-1 A simple XML document. The inventory element is the root node of this document.

Comments appear in XML documents outside of other mark-up enclosed between '`<!--`' and '`-->`' strings and are often used to give additional information to a human reader.

A *processing instruction* is enclosed within '<?' and '?>' strings and is intended to give an instruction to the application processing the document, for example, that the document should be rendered using a certain style sheet.

A *namespace* is declared using the reserved attribute name 'xmlns' and is used to uniquely define *element* and *attribute* names (Figure 2-2). A good use for *namespaces* is when merging XML content from two sources that may have conflicting *element* names or *attributes* with the same name but different semantics. Assuming different namespaces were used in the two different documents, *elements* with the same name can be distinguished.

```
<document xmlns:financial="http://foo.bar.org/financial"
xmlns:geographical="http://foo.bar.org/geographical">
  <financial:bank employees="5000" />
  <geographical:bank location="River Tyne" />
</document>
```

Figure 2-2 Example of namespaces to differentiate between elements with the same name

A *doctype* may appear at the start of a document and associates the XML document with a Document Type Definition (DTD). A DTD defines the basic structure of a document e.g. which elements are allowed, which elements an element may have as children, what the content of a particular attribute may be etc.

XML is a versatile data format with uses including web pages, databases and information exchange over HTTP. While the format is fairly verbose this comes with the advantage that semantics are explicit rather than implicit as in some other formats. As the format is typically not compressed to a binary format, the format also has the advantage of often being human understandable and being editable in standard text editing tools.

XML is employed extensively in OPSIN (Chapter 3) for encoding on-disk resources and as an in-memory representation of a parse tree. Reading XML files and manipulating in-memory representations of XML is achieved using the XOM Java XML API[39,40].

2.2 Chemical Markup Language

Chemical Markup Language (CML) [41] is the application of XML to hold chemical data. CML was developed by Professors Murray-Rust and Rzepa and initially announced in 1995[42]. Since then the format has evolved through numerous revisions and is now supported by many commercial and open source chemistry applications.

A simple use case of CML is to encode molecular structure (Figure 2-3). The elements/attributes that are allowed in CML, their allowed values for attributes and the allowed children for each element are encoded in a schema[43,44] which may be used for validation of CML[45]. If no appropriate elements are available in CML then additional information may be recorded by the use of elements or attributes outside of the CML namespace.

```
<cml xmlns="http://www.xml-cml.org/schema">
    <molecule id="m1">
        <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"/>
            <atom id="a9" elementType="H"/>
            <atom id="a10" elementType="H"/>
        </atomArray>
        <bondArray>
            <bond id="a1_a2" atomRefs2="a1 a2" order="S"/>
            <bond id="a1_a4" atomRefs2="a1 a4" order="S"/>
            <bond id="a1_a5" atomRefs2="a1 a5" order="S"/>
            <bond id="a1_a6" atomRefs2="a1 a6" order="S"/>
            <bond id="a2_a7" atomRefs2="a2 a7" order="S"/>
            <bond id="a2_a8" atomRefs2="a2 a8" order="S"/>
            <bond id="a2_a9" atomRefs2="a2 a9" order="S"/>
            <bond id="a4_a10" atomRefs2="a4 a10" order="S"/>
        </bondArray>
    </molecule>
</cml>
```

Figure 2-3 A CML document describing the connectivity of ethanol

CML has been extended to cover computational chemistry[46], spectral data[47] and polymers[48]. It has also been extended to cover chemical reactions[49]. This method of encoding chemical reactions is employed in Section 4.6.4 for serialising reactions that have been extracted from the patent literature.

2.3 SMILES

SMILES[50] (Simplified Molecular Input Line Entry Specification), published by Daylight in 1988[50], is now a widely supported convention for representing a chemical connection table as a string of ASCII text. The proliferation of applications that can read and/or write SMILES can be explained by its favourable properties compared to other contemporary line formats. These include its terseness, ease with which readers and writers can be written and that SMILES are relatively intelligible and writeable by humans.

Figure 2-4 shows a SMILES string; the string is read from left to right to generate the atoms and bonds of the molecule. The format contains many optimisations to reduce the length of the representation and improve readability; hydrogen may be implicit on organic atoms, and bonds are implicitly single or implicitly of type aromatic between aromatic atoms. Full details of the SMILES specification are available from the OpenSMILES project[51] and Daylight[52].

N[C@H](C(=O)O)Cc1ccccc1

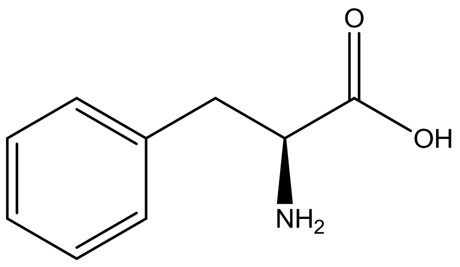


Figure 2-4 SMILES and structure for L-phenylalanine

The main limitation of SMILES is that it is not intrinsically a canonical format, meaning that a single connection table can be represented by multiple SMILES (Figure 2-5). While implementations exist that produce a canonical representation, including implementations in open source toolkits such as the CDK[53] and OpenBabel[54], no standard implementation has been agreed upon making canonical SMILES unsuitable for an interoperable canonical descriptor.

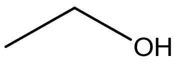
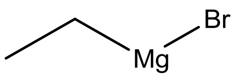
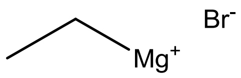


Figure 2-5 Examples of legal SMILES: CCO, OCC, C(O)C, C(C)O, CC1.O1, C1.O2.C12

Other limitations stem from the approximation of molecules as static graphs with well-defined bond orders. These include the inability to recognise that two molecules are tautomers of each other or, in the case of mesomers and bonds to metals, that two molecules are identical but simply represented differently, as exemplified in Figure 2-6. These problems, as well as the problem of having a universal canonical form are largely addressed by InChI (Section 2.4).





CC[Mg]Br CC[Mg+].[Br-]

Figure 2-6 Two representations of Ethylmagnesium bromide and their canonical SMILES (generated by OpenBabel)

SMILES are employed by OPSIN as representations for fragments of chemical names and are one of the program's output formats. They are also employed in the reaction extraction code for use as output and to allow data exchange with the Indigo toolkit[55].

2.4 InChI

InChI or IUPAC International Chemical Identifier[56] is a canonical identifier for chemical compounds. InChIs are formed of layers, in such a way that layers may be removed from right to left of the InChI without affecting the meaning of the remaining layers (Figure 2-7). This unique feature of InChI can be utilised to determine at what layer two molecules differ. For example if the InChIs of two molecules failed to match as is, but matched with the stereochemical layer removed, it can be deduced that the two molecules differ just in stereochemistry.

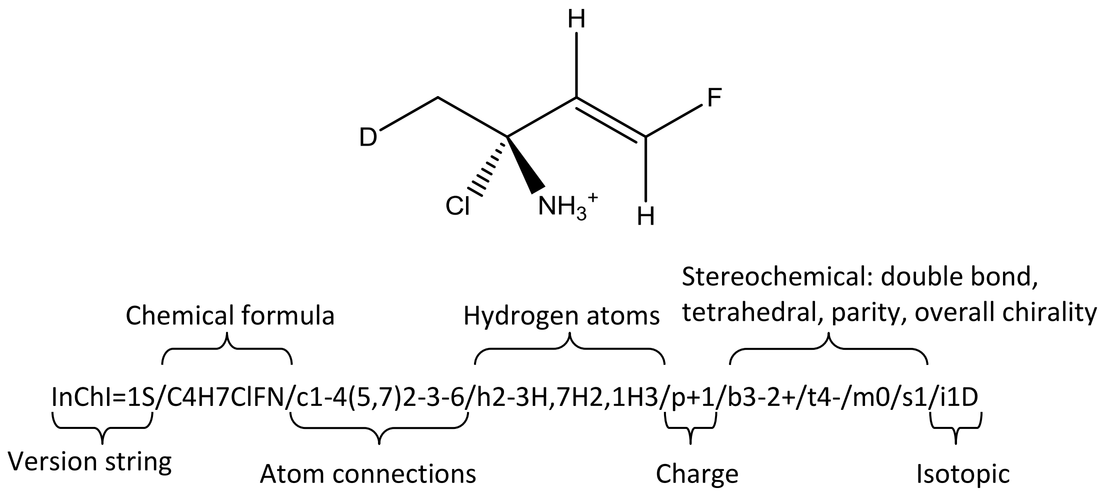
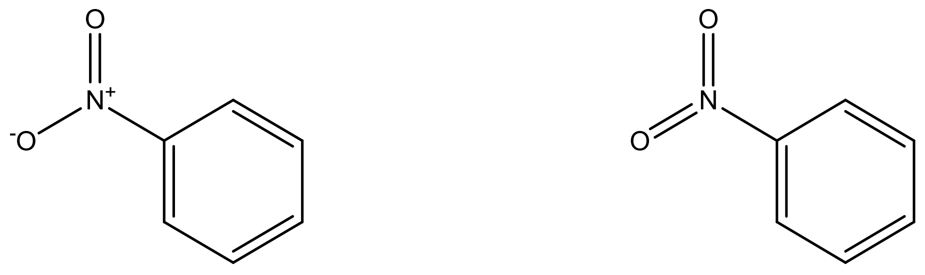

H
F
D
Cl
NH_3^+
H
Stereochemical: double bond, tetrahedral, parity, overall chirality
Chemical formula
Hydrogen atoms
InChI=1S/C4H7ClFN/c1-4(5,7)2-3-6/h2-3H,7H2,1H3/p+1/b3-2+/t4-/m0/s1/i1D
Version string
Atom connections
Charge
Isotopic

Figure 2-7 Standard InChI string with layers annotated

Unlike SMILES, InChI does not suffer from problems with the representation of organic groups for which multiple valence bond representations are possible (Figure 2-8);

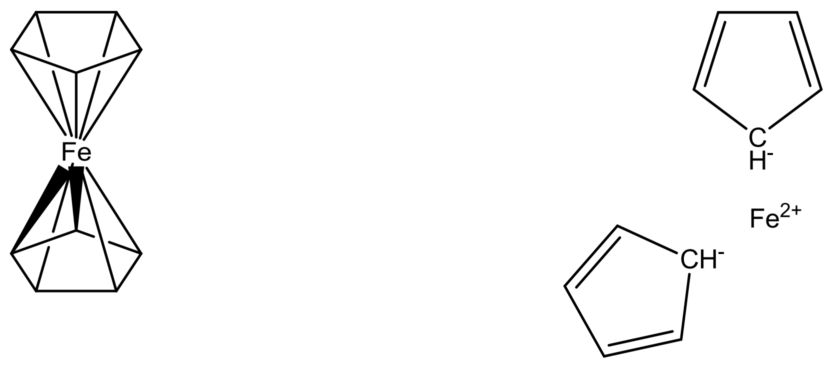


SMILES: [O-][N+](=O)c1ccccc1 SMILES: O=N(=O)c1ccccc1

InChI=1S/C6H5NO2/c8-7(9)6-4-2-1-3-5-6/h1-5H InChI=1S/C6H5NO2/c8-7(9)6-4-2-1-3-5-6/h1-5H

Figure 2-8 InChIs and SMILES for two different representations of nitrobenzene. Both representations yield the same InChI, whereas the SMILES differ.

For simple inorganic compounds the same InChI will be produced regardless of whether ionic or covalent representation is used (*cf.* Figure 2-6). However, as illustrated in Figure 2-9, multiple InChIs are possible for those inorganic compounds with bonding not found in organic compounds, such as the haptic covalent bonding between the π electrons of the cyclopentadienyl rings and the d electrons of the iron in ferrocene.



InChI=1S/2C5H5.Fe/c2*1-2-4-5-3-1;/h2*1-5H;

InChI=1S/2C5H5.Fe/c2*1-2-4-5-3-1;/h2*1-5H;/q2*-1;+2

Figure 2-9 Two possible depictions of ferrocene and the two different InChIs they produce

InChIs may be standard or non-standard, with standard InChIs being distinguished by the presence of the 'S' at the end of the version string (Figure 2-7). Unlike a standard InChI, a non-standard InChI may include a fixed-H layer that allows specification of a particular tautomer and/or a reconnected layer that explicitly includes bonds to metal atoms. A non-standard InChI may also have experimental InChI flags enabled such as those for detecting more forms of tautomerisation.

2.5 Formal grammars

A formal grammar is formed of a disjoint set of terminal and non-terminal symbols, with production rules specifying the replacements allowed for each non-terminal symbol. A terminal symbol is a literal character in the language to be recognised whilst a non-terminal symbol will have an associated production rule which defines it in terms of other terminal symbols or non-terminal symbols.

An example of a formal grammar for some simple mathematical expressions could be:

```
equation ::= bracketed-expression | expression

bracketed-expression ::= "(" expression ")"

expression ::= ( bracketed-expression | digit+ ) operator (
bracketed-expression | digit+ )

digit ::= "0" | "1" | "2" | "3"| "4" | "5" | "6" | "7" | "8" | "9"

operator ::= "+" | "-" | "×" | "÷"
```

In this grammar, each line is a production rule. The terminal symbols are the following characters: `()01233456789+−×÷` whilst the non-terminal symbols are all the other terms. A formal grammar must also have a non-terminal symbol which is designated as the start symbol, which in this case is `equation`.

The types of languages that a grammar can express are related to what restrictions, if any, are put on the allowed form of the production rules. Chomsky[57] defined four types of grammar which in order of increasing expressivity are regular, context-free, context-sensitive and unrestricted (Table 2-1).

Grammar	**Language recognised**	**Production rules**
Unrestricted	Recursively enumerable	$\alpha \rightarrow \beta$
Context-sensitive	Context-sensitive	$\alpha A\beta \rightarrow \alpha\gamma\beta$
Context-free	Context-free	$A \rightarrow \alpha$
Regular	Regular	$A \rightarrow a$ $A \rightarrow Ba$ or $A \rightarrow a$ $A \rightarrow aB$

Table 2-1 The different classes of grammars, the languages they recognise and the production rules they support. Greek symbols are any combination of terminals or non-terminal symbols, capital letters are non-terminals and lower case symbols are terminals (including the empty string).

Regular grammars and Context-free grammars will be returned to when discussing the grammar employed by OPSIN (Section 3.2.4) and ChemicalTagger (Section 2.9), respectively.

2.6 Automata

An automaton is a mathematical construct formed of states, and transitions between those states. An automaton has an alphabet which includes the symbols that may occur in the input to the automaton. An automaton of the appropriate type (Table 2-2) may be used to check that a given input is acceptable to a given grammar.

Grammar Type	Automaton type
Unrestricted	Turing machine
Context-sensitive	Linear bounded automaton
Context-free	Pushdown automaton
Regular	Finite state automaton

Table 2-2 Automaton types required to process the archetypal grammar types

As this work only employs regular and context-free grammars this exposition will focus on the properties of the corresponding automata for these grammars.

A finite state automaton (FSA) is the simplest automaton and is demonstrated by the example shown in Figure 2-10. Each circle is a state and every arrow is a transition. When the automaton is "run" over a given input, the input is consumed character by character with an appropriate transition being attempted after consumption of each character. The character that must be consumed for a transition to be allowed is indicated next to the arrow. If no appropriate transition is possible then the input is not acceptable to the grammar. The automaton continues through the input until all characters have been consumed at which point examination of whether the FSA is in an accept state (double circle in diagram) indicates whether the input was accepted by the grammar.

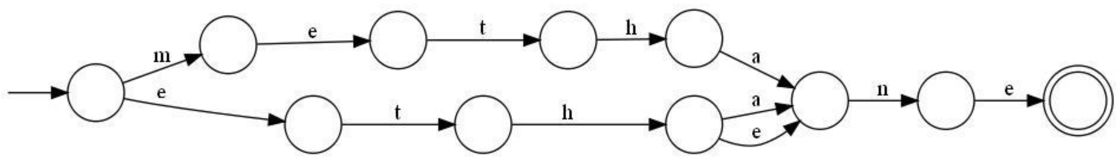


Figure 2-10 A finite state automaton that matches "methane", "ethane" and "ethene"

A pushdown automaton is the same as a FSA but with the exception of also having a stack. The top of this stack may be inspected to determine which transition to make and as part of performing a transition an entry may be added or removed from the stack. Figure 2-11 demonstrates a context-free grammar and Figure 2-12 shows a pushdown automaton that could describe it.

```
expression = bracketed-expression | "a" ;

bracketed-expression = "(" , ( bracketed-expression | "a" ) , ")" ;
```

Figure 2-11 A context-free grammar; a stack is required to keep track of the nesting

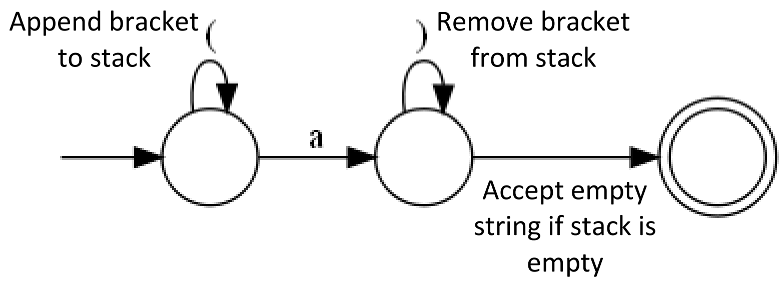

Append bracket to stack
(
)
Remove bracket from stack
a
Accept empty string if stack is empty

Figure 2-12 A possible state machine for the grammar from Figure 2-11

2.7 Regular expressions

Regular expressions are a widely supported method of encoding patterns for finding strings. A true regular expression can always be expressed using a regular grammar (*cf.* Section 2.5) and the expression it describes can be matched by a finite state automaton (*cf.* Section 2.6). Due to the widespread usage of Perl-esque regular expressions, which are capable of matching languages that are less restrictive than even context-free languages, it is useful to draw a distinction between true regular expressions and "regexes". Regexes may contain operators that necessitate such operations as look ahead, look behind and references to named capture groups.

At its simplest, a regular expression is just the string for which one wishes to search. Metacharacters may be used to achieve more expressive searches (Table 2-3). To match the literal metacharacter the character is preceded by a forward slash. Forward slashes also proceed abbreviated character classes e.g. \d for digits, \D for non-digits, \s for whitespace and \S for non-whitespace.

Metacharacter/s	Meaning
.	Match any character
[ ]	Mark the start and end of the description for a single character e.g. [ab] is either 'a' or 'b'; [a-z] is any of the 24 lower case characters
[^]	As above but matches a character that does not meet the description
^	Start of string
$	End of string
( )	Demarcate a sub expression. In regexes this is by default also a capturing group
*	Indicates the preceding expression should be repeated 0 or more times
?	Indicates the preceding expression is optional
+	Indicates the preceding expression should be repeated 1 or more times
{m,n}	Indicates a range of number of times the preceding expression should be repeated
|	Indicates a choice between the expressions either side of the operator
\	Used to indicate a literal metacharacter or a shorthand character class

Table 2-3 Regular expression metacharacters

Regular expressions are used as the input to build OPSIN's grammar, describing the form of systematic chemical names. Regexes are employed in various places throughout all projects.

2.8 OSCAR4

OSCAR (Open Source Chemistry Analysis Routines) began as a tool for checking experimental data[58]. The library, now in its fourth major revision, contains functionality for extracting chemical entities from free text as well as, where possible resolving them to structures or ontology identifiers (e.g. ChEBI ids[5]). The functionality for identifying and interpreting experimental data sections is also retained.

As OSCAR has developed, so have the algorithms employed. This is especially pronounced in the area of identifying chemical names. The original dictionary lookup approach was supplemented by the addition of heuristic identification of chemical entities through regular expressions[59], N-gram analysis[60] and finally by a maximum-entropy Markov model (MEMM)[9]. In this context, N-gram analysis refers to calculating a probability that a word is chemical from the analysis of occurrences of the constituent one to four letter sequences in the word as compared to known occurrences in a training set of chemical and non-chemical words. A MEMM is used to predict the labels for a sequence, in this case a sequence of tokens. OSCAR has a separate MEMM model for each of the entity types that are not found by string matching. These entity types are chemical, reaction e.g. hydroxylation, chemical adjective and enzyme. Features employed by the MEMM models include 1-4 character N-grams, the suffix of the token, whether it appears in any word lists e.g. English words, and adjacent tokens to a given token.

The current version of OSCAR is OSCAR4[14] which differs from previous incarnations by being divided into modules using Maven (*cf.* section 2.10 and Figure 2-13). This allows for independent aspects of the program to be utilised without bringing in the entirety of OSCAR4.

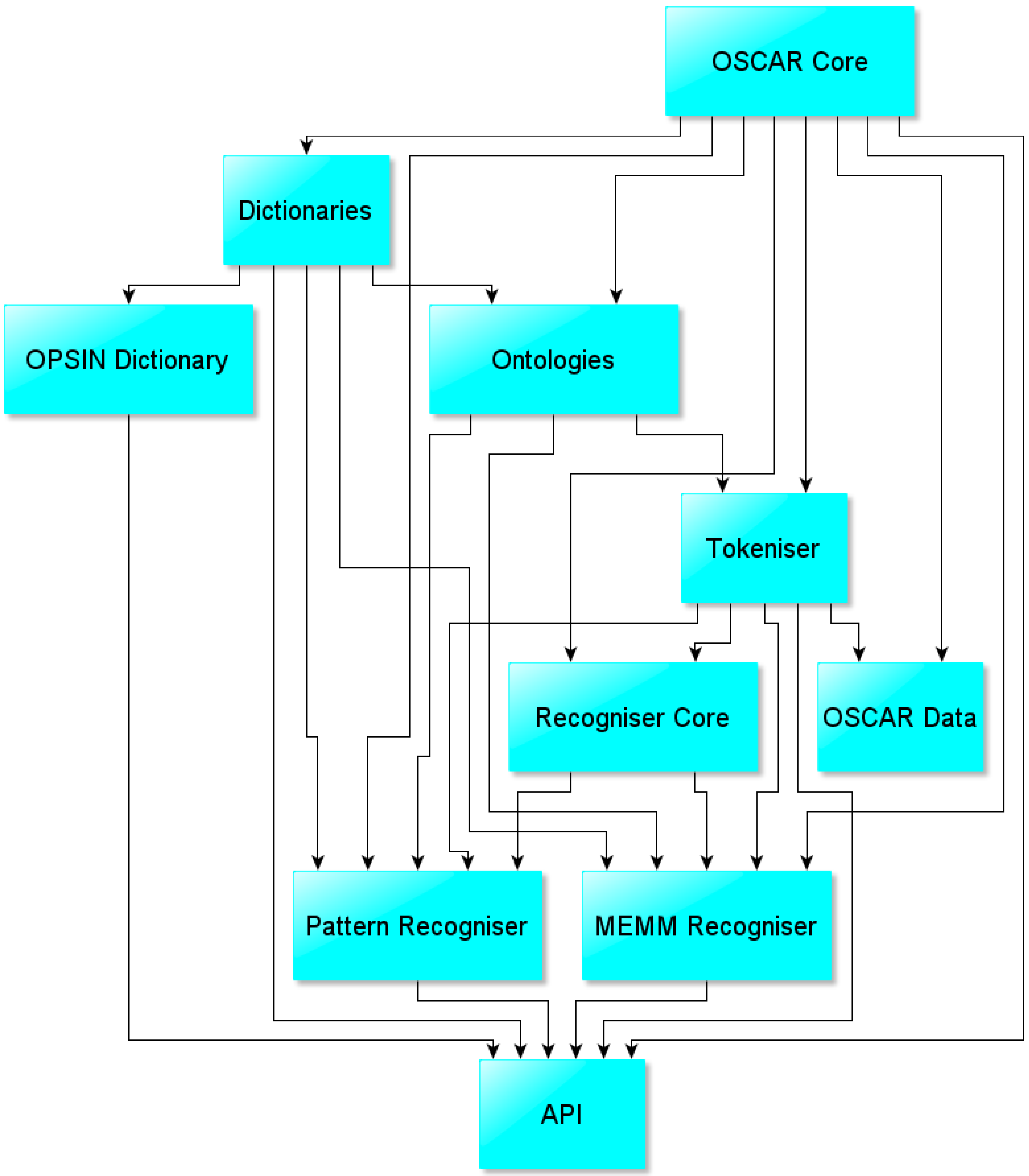


Figure 2-13 Interdependencies between the modules of OSCAR4

OSCAR4 is indirectly employed in this work as a tagger and tokeniser for use in ChemicalTagger (Section 2.9). To improve ChemicalTagger's performance some improvements were made to OSCAR4's tokeniser (Section 4.4.5.1). Input was also put into developing the OPSIN dictionary which is an implementation of OSCAR4's `IChemNameDict` interface backed by OPSIN.

2.9 ChemicalTagger

ChemicalTagger[61] is a tool for annotating chemical text that was developed in the Murray-Rust group of the Unilever Centre, Cambridge. Its overall function is to attempt to extract semantic information from chemistry documents and, in particular, from experimental sections. Figure 2-14 gives a schematic of the program's architecture.

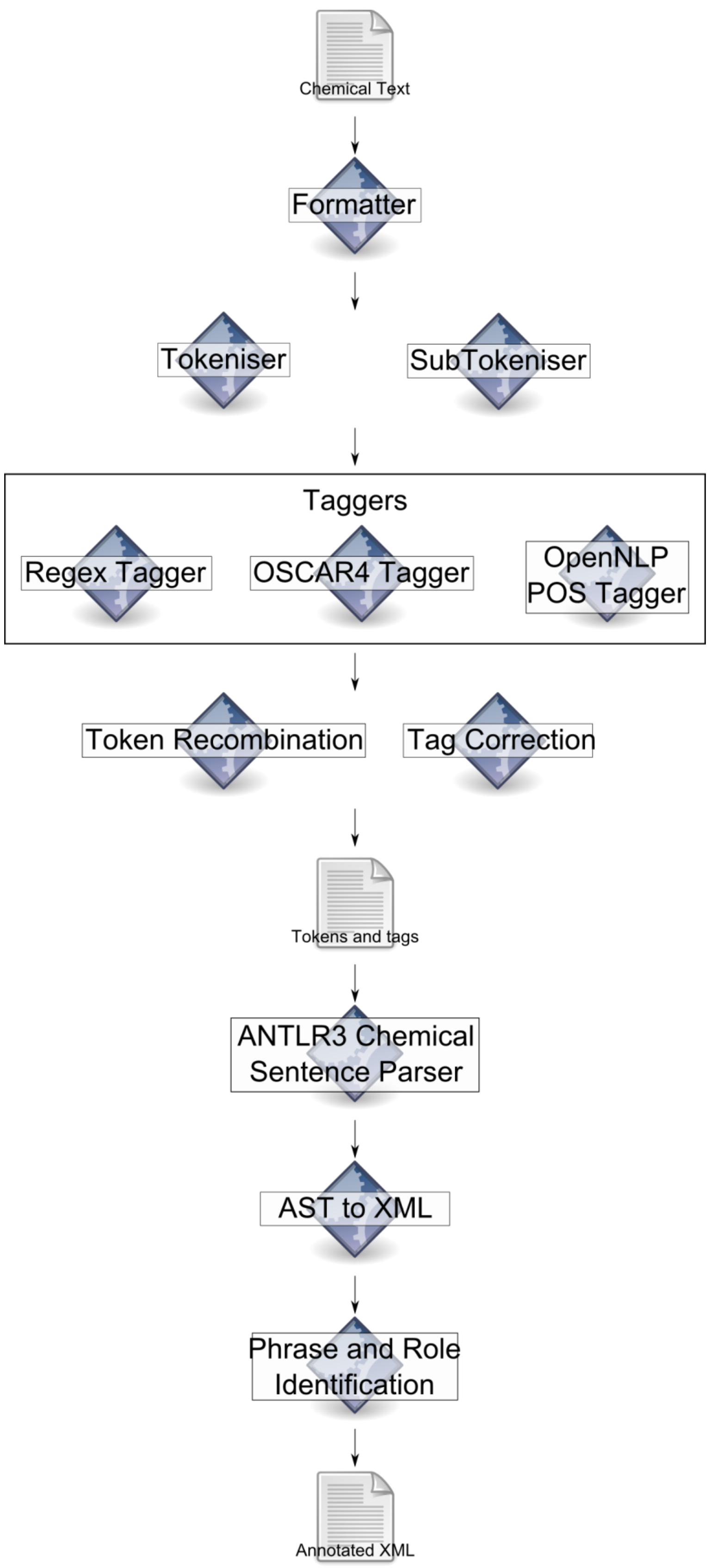


Figure 2-14 Architecture of ChemicalTagger

The input to the program is a string of text, typically a paragraph. The first step in the workflow involves the `Formatter` normalising the input. For example, all hyphens are normalised to a single hyphen type.

The next step is tokenisation. ChemicalTagger has a tokenisation interface which is implemented by both an OSCAR4 tokeniser and a simpler whitespace tokeniser. For synthetic chemistry text, the OSCAR4 tokeniser typically performs better and hence was used in this work.

Some specific tokenisations that are unlikely to be performed by more general tokenisers are performed by the sub-tokeniser, the most important of which is the tokenising of numbers directly concatenated to a unit. For example '50ml' is tokenised to ['50', 'ml']. This is necessary to allow such cases to be recognised as two tokens, hence allowing the numeric value and unit to be separately tagged.

The tokenised input is then passed through a series of taggers which implement a tagger interface allowing easy addition of more taggers. By default, the system employs a regex tagger, an OSCAR4 tagger and a part of speech (POS) tagger, provided by OpenNLP[62].

The regex tagger identifies chemistry related terms. For example, verbs relating to chemical processes such as 'heated', and adjectives relating to chemicals e.g. 'anhydrous'. For greater specificity later in the workflow, some prepositions are explicitly tagged, e.g. 'in' has its own tag. The OSCAR4 tagger tags chemicals, as well as some enzymes, chemical adjectives and reaction adjectives. The POS tagger tags all tokens with a POS tag using the Penn Tree bank POS tags[63]. For the cases where the POS tag is the literal character, the tag is changed e.g. '.' becomes 'STOP'. The results from the taggers are then combined using a user tuneable order of preference as a decider in the case where more than one tagger produces a tag for a token.

A limitation that is quickly encountered with naive regex tagging of key words is polysemy, the capacity for the same word to have multiple meanings dependent on context. One of the more common and important distinctions that needs to be made is between a word being a verb or an adjective e.g. 'the solution was **concentrated** using' as compared to '**concentrated** sulfuric acid'. These problems, as well as a few special cases indicative of excessive tokenisation, are identified and corrected by hand crafted rules, prior to producing the final lists of tokens and tags. Table 2-4 shows an example input after tokenisation and then after tagging.

Input	To a vigorously stirred solution of pyridine in THF (40mL)											
Tokenised	To	a	vigorously	stirred	solution	of	pyridine	in	THF	(	40	mL	)
Tagged	**TO**	DT	RB	**JJ-CHEM**	**NN-CHEMENTITY**	**IN-OF**	*OSCAR-CM*	**IN-IN**	*OSCAR-CM*	**-LRB-**	**CD**	**NN-VOL**	**-RRB-**

Table 2-4 Example of ChemicalTagger output after tokenisation and output after tagging. Bold terms are identified by the regex tagger, italicised by the OSCAR4 tagger and the remaining two tokens by the POS tagger. Note that 'stirred' is initially tagged as a VB-STIR before subsequently being corrected, during tag correction, to a JJ-CHEM due to its adjacency to an NN-CHEMENTITY.

The lists of tags and tokens are then interlaced and given to the chemical sentence parser. This parser is prebuilt from an ANTLR3[64] grammar. This grammar has been hand-written to describe the makeup of experimental chemistry paragraphs. ANTLR3 grammars nominally describe a subset of context-free grammars but may also include "semantic predicates". A semantic predicate allows the execution of a Boolean method to determine whether or not a production rule in the grammar may be used. Such a method may examine the current token or any previous or future token and execute arbitrary code to make its decision. In principle, the method could even have persistent state allowing the parsing of languages requiring greater expressivity than a context-free grammar.

The chemical sentence parser will attempt to break the input down into a tree structure called an abstract syntax tree (AST). This is formed of sentences which in turn are formed of phrases e.g. "NounPhrase", "VerbPhrase", "PrepPhrase". Phrases are formed of other phrases, compound constructs such as "MOLECULE"s or of tags. At its deepest level the tree will be formed entirely of tags which correspond to the tags originally input to the sentence parser.

The AST is subsequently converted to XML and enriched by the annotation of phrase types and roles for some molecules. Assignment of phrases is achieved by looking for the presence of certain tags. For example a phrase would be annotated as a "Yield" if it contained a VB-Yield. Solvents are identified by their presence after key words when in certain phrases, such as "Dissolve" and "Wash" phrases. Roles may also be assigned from a successful match with the Hearst pattern[65] [MOLECULE] 'as a' [NN-CHEMENTITY], or from a molecule being mentioned as being 'in' another molecule, indicating the latter molecule to likely be a solvent. Figure 2-15 shows an example of the final XML output.

```
<Document>
  <Sentence>
    <PrepPhrase>
      <TO>To</TO>
      <NounPhrase>
        <DT>a</DT>
        <RB>vigorously</RB>
        <JJ-CHEM>stirred</JJ-CHEM>
        <NN-CHEMENTITY>solution</NN-CHEMENTITY>
        <PrepPhrase>
          <IN-OF>of</IN-OF>
          <NounPhrase>
            <ActionPhrase type="Dissolve">
              <MOLECULE>
                <OSCARCM>
                  <OSCAR-CM>pyridine</OSCAR-CM>
                </OSCARCM>
              </MOLECULE>
              <IN-IN>in</IN-IN>
              <MOLECULE role="Solvent">
                <OSCARCM>
                  <OSCAR-CM>THF</OSCAR-CM>
                </OSCARCM>
                <QUANTITY>
                  <_-LRB->(</_-LRB->
                  <VOLUME>
                    <CD>40</CD>
                    <NN-VOL>mL</NN-VOL>
                  </VOLUME>
                  <_-RRB->)</_-RRB->
                </QUANTITY>
              </MOLECULE>
            </ActionPhrase>
          </NounPhrase>
        </PrepPhrase>
      </NounPhrase>
    </PrepPhrase>
  </Sentence>
</Document>
```

Figure 2-15 Final ChemicalTagger output for the input from Table 2-4.

The output from ChemicalTagger is vital to many aspects of the reaction extraction system described in Chapter 4. As a result significant effort was made, during this project, to improve ChemicalTagger's performance and extend it to identify concepts of importance for extracting reactions as further described in Section 4.4.5.

2.10 Apache Maven

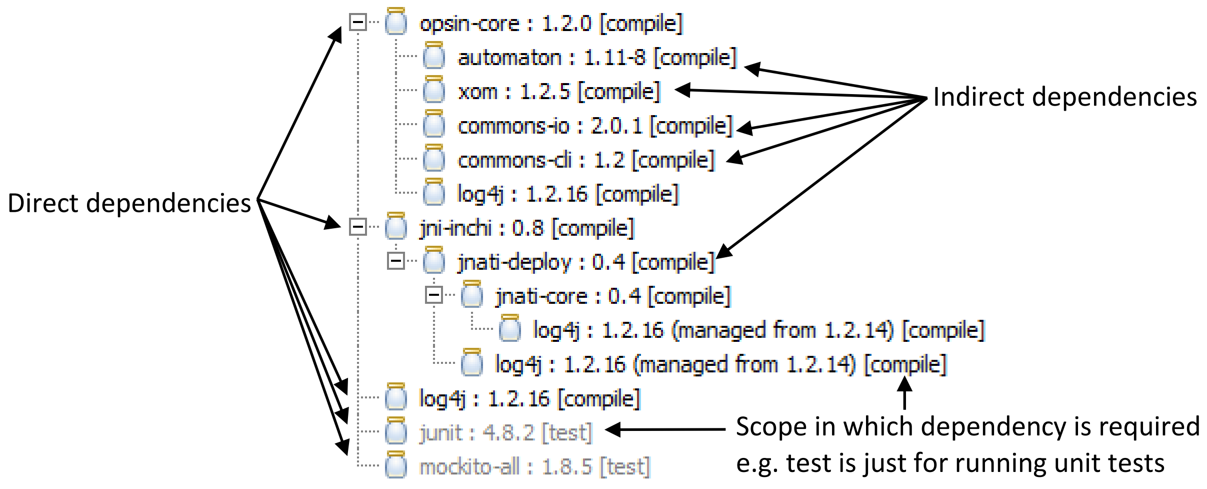
Science often works by building on prior achievements and the same is true of software development. When building more advanced software, it is typically easier and quicker to employ existing solutions to problems as dependencies rather than attempting to re-implement their functionality. As the number of dependencies of a project increases, managing these dependencies can become time consuming. The Apache Maven build system[66] offers numerous advantages, especially in managing larger projects with many dependencies.

The system works by each project corresponding to an artifact, or multiple artifacts in the case of a project made from multiple modules. Each artifact is assigned a groupId, artifactId and version description e.g.

```
<artifactId>chemicalTagger</artifactId>
<groupId>uk.ac.cam.ch</groupId>
<version>1.3.1</version>
```

The artifactId is the name of the project/module and the groupId is a unique string which is typically a domain name that you control. The combination of these fields should be sufficient to uniquely identify a particular artifact. For released artifacts, the version number will typically be numeric and for a given artifactId/groupId must be unique. For rapid development, it is often useful for upstream projects to be able to depend on the latest version of a dependency without the need to constantly update the dependency version. This can be accomplished by adding the special suffix '–SNAPSHOT' to the version description string of the artifact. As a snapshot version of an artifact changes with time, snapshot versions of dependencies should not be relied upon for releases.

Artifacts are stored in Maven repositories which, typically, are internet accessible. From these repositories, the artifacts that a project requests are downloaded for local use. These dependencies may in turn have their own dependencies which are recursively acquired. Figure 2-16 shows an example of the result of this for the InChI module of OPSIN.


Indirect dependencies
Direct dependencies
Scope in which dependency is required
e.g. test is just for running unit tests

```
opsin-core : 1.2.0 [compile]
    automaton : 1.11-8 [compile]
    xom : 1.2.5 [compile]
    commons-io : 2.0.1 [compile]
    commons-cli : 1.2 [compile]
    log4j : 1.2.16 [compile]
jni-inchi : 0.8 [compile]
    jnati-deploy : 0.4 [compile]
        jnati-core : 0.4 [compile]
            log4j : 1.2.16 (managed from 1.2.14) [compile]
        log4j : 1.2.16 (managed from 1.2.14) [compile]
log4j : 1.2.16 [compile]
junit : 4.8.2 [test]
mockito-all : 1.8.5 [test]
```

Figure 2-16 Dependency hierarchy for the opsin-inchi module

All the projects involved in this work were either natively available via Maven, or were manually added to our Maven repository. As can be seen from Figure 2-17, advanced projects can easily require a non-trivial number of dependencies.

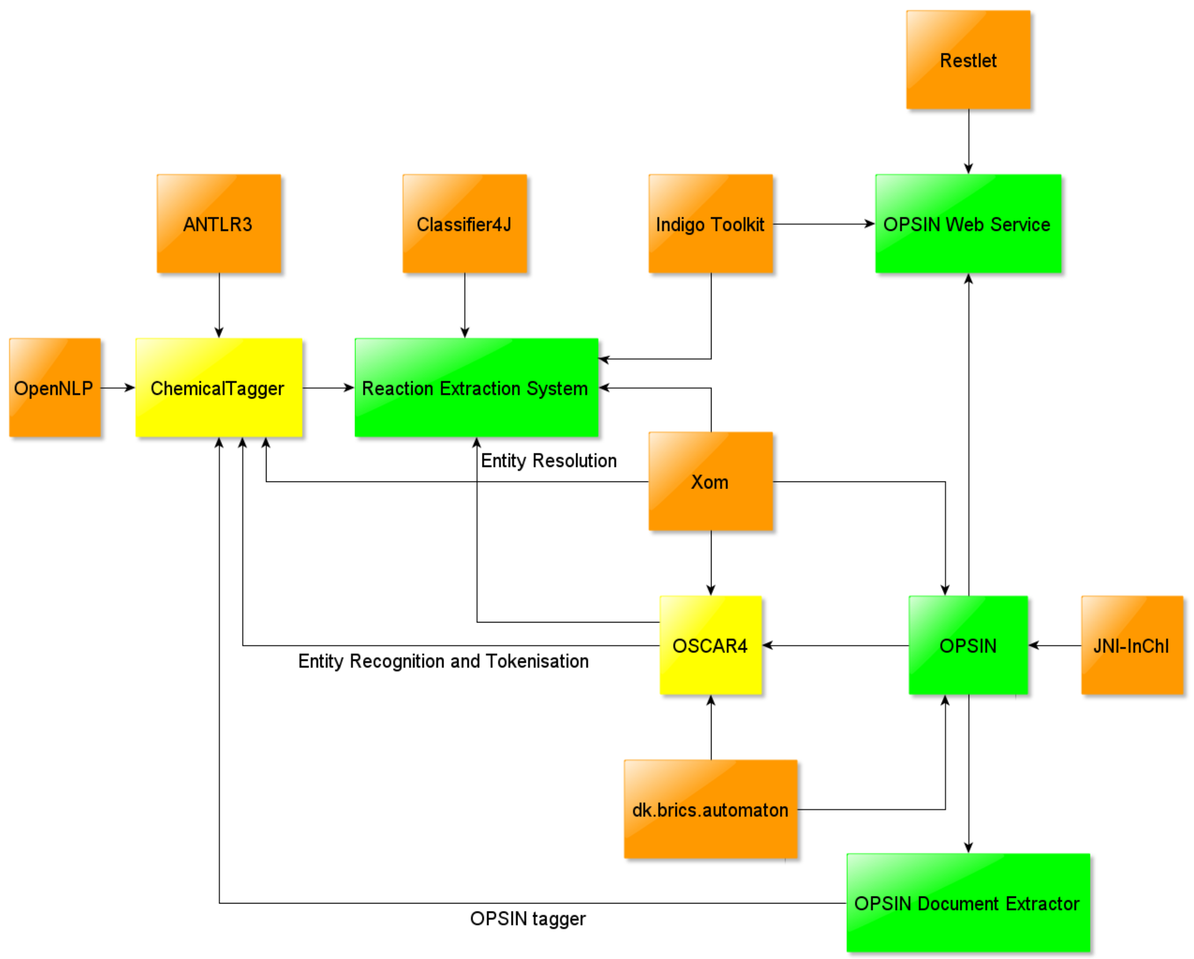


Figure 2-17: How the open source projects involved in this project relate to each other. Green projects are projects developed predominantly for this project whilst yellow projects are those in which significant improvements were undertaken. Utility and unit testing libraries are not shown.

2.11 Distributed version control

In larger software development projects, such as the ones involved in this work, version control is essential. Version control allows a developer to associate a message with each set of changes that are made to a project's code or dependent files. Reverting the state of particular files or the whole project to an older revision is then trivial and is useful when one needs to investigate the effect that a particular change had on a program's output or to reinstate previously removed functionality. On larger projects, the version control system must support multiple developers committing changes, to which an elegant solution is a distributed version control system.

Under a distributed version control system each developer has a local repository into which their changes are committed. A notable advantage of this approach is that significant changes to a program can be done incrementally and only distributed when the program is once again stable.

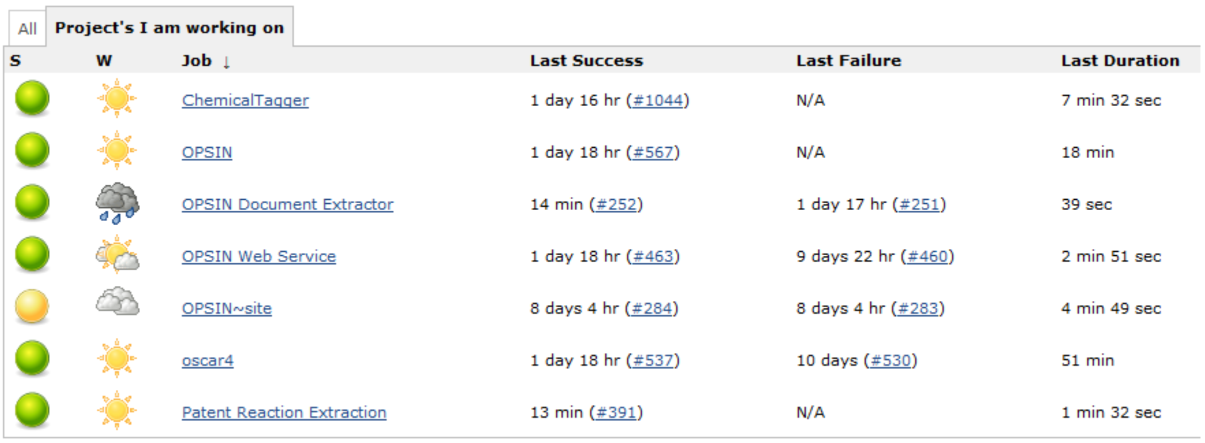
Change sets can be transferred between repositories by “pushing” or “pulling”. For accessibility, backup purposes and for clarity as to what is the latest code, it is useful to have a central repository on a web-based site such as Bitbucket[67] or GitHub[68] that is readily accessible to all involved. It should be noted that such a repository is only the central repository by convention as it does not differ in structure from any of the other repositories.

The software developed as part of this thesis employs Mercurial[69] for version control, due to its wide support and ease of use, and Bitbucket for code hosting.

2.12 Continuous integration testing

As projects get larger it is highly useful to be able to define tests indicating the expected output of methods for a given input. These can be used to assure that changes have not broken existing functionality and that added functionality is functioning as expected. When coding in Java, an easy way of implementing such tests is by using JUnit[70]. A continuous integration service can be set up to constantly, or periodically, query a repository for changes. If a change has been committed, the service automatically builds the project and runs its unit tests. If a failure in building the project or running its unit tests is detected, the developers of the project can be emailed allowing them to immediately look into the cause of the failure. The Jenkins[71] continuous integration service was used for this purpose (Figure 2-18).

For Maven projects, Jenkins can be configured to automatically deploy snapshot versions of the project to a Maven repository allowing the project’s dependents to instantly benefit from the updated version. Additionally, Maven projects that Jenkins is aware of, which depend on such a project can be automatically built and tested to check that the updated dependency has not caused problems in any of these upstream projects. For example, whenever OSCAR4 is updated, OSCAR4 will be built and tested. The same will then happen for ChemicalTagger, and then the patent reaction extraction project, as each depends on the previous project.


All | **Project's I am working on**

S	W	Job ↓	Last Success	Last Failure	Last Duration
🟢	☀	ChemicalTagger	1 day 16 hr (#1044)	N/A	7 min 32 sec
🟢	☀	OPSIN	1 day 18 hr (#567)	N/A	18 min
🟢	🌧	OPSIN Document Extractor	14 min (#252)	1 day 17 hr (#251)	39 sec
🟢	⛅	OPSIN Web Service	1 day 18 hr (#463)	9 days 22 hr (#460)	2 min 51 sec
🟡	☁	OPSIN~site	8 days 4 hr (#284)	8 days 4 hr (#283)	4 min 49 sec
🟢	☀	oscar4	1 day 18 hr (#537)	10 days (#530)	51 min
🟢	☀	Patent Reaction Extraction	13 min (#391)	N/A	1 min 32 sec

Figure 2-18 View of selected projects from Jenkins. The coloured orb indicates at a glance whether the project's last build was successful (green), had unit test failures (yellow) or failed (red). The "weather" pictogram indicates the stability of previous builds.

Chapter 3 Conversion of Chemical Names to Structures

This chapter describes the work undertaken in this project on the successful development of OPSIN (Open Parser for Systematic IUPAC Nomenclature) an open source chemical name to structure algorithm. A paper based on the work more fully described in this chapter was published in the Journal of Chemical Information and Modelling[72]. The paper was the journal's most accessed paper during the month it was published and was among the top 20 most accessed for the year in March 2012. OPSIN was also included in a review of open source cheminformatics applications[73].

3.1 Introduction

3.1.1 History of systematic nomenclature

Compared to most spoken languages systematic chemical nomenclature is a relatively recent invention with initial codification at the 1892 Geneva conference[74]. This conference defined a system of nomenclature, known as the Geneva system, allowing the specification of simple compounds (e.g. hydrocarbons) with substituents and common functional groups. This system evolved through multiple recommendations[75–80] into what is now known as IUPAC nomenclature. The interested reader is referred to Smith Jr.'s review paper documenting the history of systematic organic nomenclature[81].

The Chemical Abstracts Service (CAS)[82,83] and Beilstein have also developed systematic nomenclature for use in Chemical Abstracts and Beilstein's Handbook of Organic Chemistry respectively. These nomenclature systems employ for the most part the same vocabulary and operations as IUPAC nomenclature but are designed to uniquely assign a name to each compound. As a result the nomenclature operations they describe are approximately a subset of those documented by the IUPAC.

An additional consideration when producing an alphabetical listing of chemicals is that structurally related compounds should be listed closely together. This is achieved in CAS nomenclature through the use of inverted index names which place the parent group in front of the group's substituents e.g. '4-aminobenzenesulfonamide' becomes 'benzenesulfonamide, 4-amino-'. Traditionally some substitutions of common parent groups yielded different trivial names which would likely end up far apart in the index. To alleviate this problem, CAS index names have become steadily more systematic by the removal of trivial names in favour of systematic parent group names (Figure 3-1).

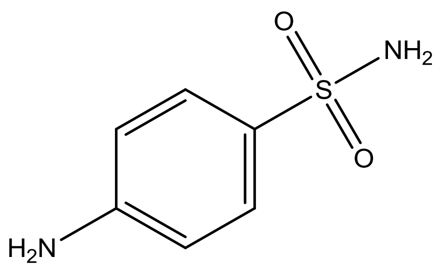


Figure 3-1 CAS index name: Benzenesulfonamide, 4-amino- (trivial name: sulfanilamide)

3.1.1 Classes of chemical name

Chemical names may be broadly categorised as systematic, semi-systematic and trivial (Figure 3-2). A trivial name cannot be decomposed into morphemes and can only be understood by dictionary lookup. If a trivial name is substituted in a logical manner then the name is semi-systematic. Names in which the parent group and all substituents are named in such a way that their structures may be deduced by breaking them down into morphemes are systematic.

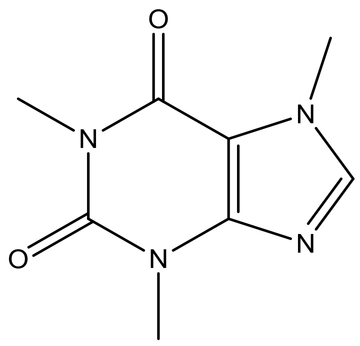


Figure 3-2 1,3,7-trimethyl-1*H*-purine-2,6(3*H*,7*H*)-dione (systematic), 1,3,7-trimethylxanthine (semi-systematic), caffeine (trivial)

This categorisation is made far more blurry by the presence of retained trivial names in IUPAC nomenclature. These are groups that are preferred over their systematic alternatives (e.g. 'purine') or even in some cases the only allowed option. For example the alkane chains of length less than five e.g. methane and ethane are trivial as they do not start with a morpheme indicating the number of carbons in the chain. Names such as 'monane' and 'diane' would be systematic but are unknown.

3.1.2 General construction of systematic names

The order of construction of a systematic substitutive name is outlined in Figure 3-3.

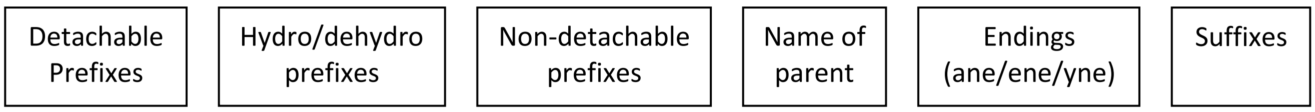


Figure 3-3 Components of a substitutive name

- Detachable prefixes:

  These are terms such as 'ethyl', 'chloro' etc. They describe groups that will be attached to the parent compound. Formally these describe radicals and are referred to as substituents in this text.

- Hydro/dehydro prefixes:

  E.g 'dihydro'. These describe the addition or removal of hydrogen from a system. They are used inconsistently as if they were a detachable or non-detachable prefix.

- Non-detachable prefixes:

  E.g. 'aza', '1H-', 'cyclo', 'methano', 'benzo' etc. These prefixes are used to modify the parent group e.g. changing atom element type, cyclising the structure, adding bridges etc.

- Name of parent:

  E.g. 'meth', 'benzene', 'acet' etc. This is the name of the parent group.

- Endings:

  These are employed on alkanes and some natural products to indicate unsaturation e.g. 'ene'.

- Suffixes:

  There are two types of suffixes, cumulative suffixes that may be used in combination with other suffixes e.g. 'ium', 'yl' and functional suffixes that describe the functionality of the group and only one of which may be present e.g. 'amide', 'oic acid'.

  Other components that may appear in multiple parts of a chemical name include:

- Locants:

  These may be numeric, Greek characters or an element symbol (which may be used in conjunction with a non-element symbol locant). Locants indicate the position on the parent group referred to by the operation that the locant precedes.

- Multipliers:

  These indicate that an operation should be performed multiple times.

- Stereodescriptors:

  These are used to indicate the stereochemistry of a detachable prefix or parent group.

3.1.3 History of programmatic name to structure conversion

Efforts to employ computers to extract information from chemical names dates back to the work of Garfield in 1962[84,85]. His program decomposed simple substitutive chemical names, formed of acyclic components, into their composite morphemes. Each morpheme can then be treating as either specifying a molecular formula e.g. 'prop' = C_3 or modifying the molecular formula e.g. 'ene' indicates the presence of a double bond. A relatively simple formula was then employed to calculate the hydrogen count from the heavy atom composition and the number of double bonds, hence yielding a molecular formula. A molecular formula could then be used as a search parameter, for example to search for compounds of identical composition in formula indexes of resources such as Chemical Abstracts.

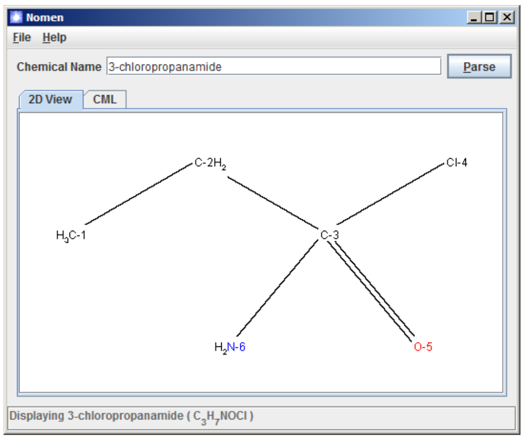
CAS published in 1967 and 1974[86,87] on an in-house tool for converting chemical names in CAS nomenclature to structures. The aims were to verify that the names were syntactically valid and ultimately that they agreed with the structure that was present in the CAS registry. The approach interpreted chemical names in a left to right manner, without the use of a formal grammar. The program is documented as supporting fused ring nomenclature (via dictionary lookup), bridges, hydro prefixes, prefix functional replacement of oxygen by sulfur, von Baeyer nomenclature, spiro nomenclature, conjunctive nomenclature, skeletal replacement, ring assemblies of two rings, and special cases of subtractive nomenclature. As the inverted index form of chemical names is most often used in CAS nomenclature, this tool is able to handle uninverted names through a special case that inverts them.

In the 1980s, work at the University of Hull by Kirby et al. yielded a name to structure algorithm based upon a formal grammar[88–93]. Their solution parsed chemical names from right to left using a context-free grammar. The series of publications noted that technically IUPAC nomenclature is a context-sensitive language if one enforces the order of enclosing marks (from outermost to innermost: "{", "[", "("), but if one does not enforce this order of bracket nesting the language is then context-free. These publications also acknowledged that, beyond bracketing, much of IUPAC nomenclature can be expressed by a regular grammar; which is the approach that is utilised by OPSIN (Section 3.2.4).

Another interesting solution was CHEMNAME, developed by Chugai Pharmaceuticals in Japan. This solution has been published through a series of seven Japanese conference papers from 1991-2005. All these papers are written in Japanese making it difficult for non-Japanese readers to understand the details of the solutions and algorithms described. The most recent papers in the series describe advanced capabilities such as algorithmic handling of fused ring systems and support for natural product nomenclature[94–96].

3.1.4 Current solutions

At the time of starting this project in 2008, two open source attempts at performing chemical name to structure were identified: ChemNomParse[97] from the University of Manchester and OPSIN[60] from the University of Cambridge. ChemNomParse had not been updated since 2003 and in testing was found to only support alkanes, cycloalkanes, simple substitution from common substituents and common suffixes. Precision was also found to be poor primarily due to terminal suffixes being systematically misplaced (Figure 3-4).



Chemical Name 3-chloropropanamide

Displaying 3-chloropropanamide (C_3H_7NOCl)

Figure 3-4 Output for 3-chloropropanamide from the ChemNomParse GUI. The suffix is placed at the end of the chain rather than the beginning.

OPSIN (Open Parser for Systematic IUPAC nomenclature) was a component of OSCAR3, a tool for performing chemical text mining. The program supported alkanes, unsaturation and cyclisation of alkanes, most Hantzsch-Widman nomenclature, bicyclo von Baeyer systems, mono spiro systems with two rings, substitutive nomenclature, common suffixes, hydro and indicated hydrogen prefixes and multi word names for esters/acyl halides/salts.

A review of the area[98] gave the following contemporary description of OPSIN:

"*OPSIN is presently limited to the decoding of basic IUPAC nomenclature but can handle bicyclic systems, and saturated heterocycles. OPSIN does not currently deal with stereochemistry, organometallics and many other expected domains of nomenclature*"

Commercial solutions are available from ACD/Labs[99], Bio-Rad Laboratories[100], PerkinElmer (formerly CambridgeSoft)[101], ChemAxon[102], ChemInnovation[103], InfoChem[104] and OpenEye[105]. With the exception of PerkinElmer's solution none of these approaches have been detailed in the literature.

PerkinElmer's solution, Name=Struct[106], takes a lenient approach to chemical nomenclature with the intention of supporting not just well formed names but names with minor mistakes, names that explicitly contradict nomenclature recommendations and even names conforming to no formal nomenclature recommendations. To this end Name=Struct takes a more relaxed approach to tokenisation choosing to always recognise the longest allowed token at each character with ad hoc rules preventing incorrect tokenisations and any punctuation being allowed to delimit tokens. Tokens are associated with one or more meanings ordered in a hierarchy of preference with disambiguation being achieved by examination of the token's local environment. This process is described in detail in a patent from CambridgeSoft[107]. The Name=Struct algorithm is incorporated into ChemBioDraw.

MDL have a European patent that covers software to extract chemical information and in particular reactions from text[108]. The patent describes a software application named Reverse AutoNom. This software does not appear to be commercially available to the public.

The Heidelberg Institute for Theoretical Studies (formerly European Media Laboratories) has also investigated this problem and produced CLP(name2structure)[109]. This application differs somewhat from the previous solutions as it aims to be able to represent ambiguous chemical names.

This software is not currently distributed and it is unclear from the paper where the described system lies between a proof of concept and a comprehensive solution.

3.2 Development and implementation of OPSIN

3.2.1 Strategy for development of OPSIN

The current version of OPSIN was arrived at by the incremental addition of areas of nomenclature. The most important aspect when adding support for a new area of nomenclature was to make sure that as new names became parsable that they either produced the intended interpretation or the case was recognised as not being currently supported and hence no structure was returned. In this way new nomenclature can be added whilst maintaining high precision.

As new nomenclature often required new functionality e.g. the ability to specify stereochemistry, the underlying capabilities of the program were also incrementally updated. As much nomenclature was not considered in the program's original design, refactoring of existing functionality was frequently required to provide a framework that could elegantly support both the added and existing nomenclature.

The version of OPSIN documented in this thesis ultimately shares very little in common with the version of OPSIN inherited in 2008. The current codebase of approximately 27,000 lines of Java is nearly an order of magnitude larger and even areas of nomenclature nominally supported by the original program have been overhauled to allow more complete support.

All subsequent references to OPSIN refer to version 1.2.0. This is latest released version as of the time of writing and was released on the 6th December 2011. Unless explicitly stated to the contrary all chemical names and nomenclature mentioned are supported and correctly interpretable. As OPSIN is designed to be employed on real world names, not all names given as exemplars strictly conform to IUPAC recommendations. Depictions for exemplars are generated by the ChemBioDraw12 using SMILES produced by OPSIN.

3.2.2 Architecture

OPSIN is written in Java with grammar and token definitions described in XML. A schematic of OPSIN's workflow is presented in Figure 3-5. The following subsections describe and discuss the implementation of the components of this workflow, with specific examples used to exemplify the types of nomenclature that are handled.

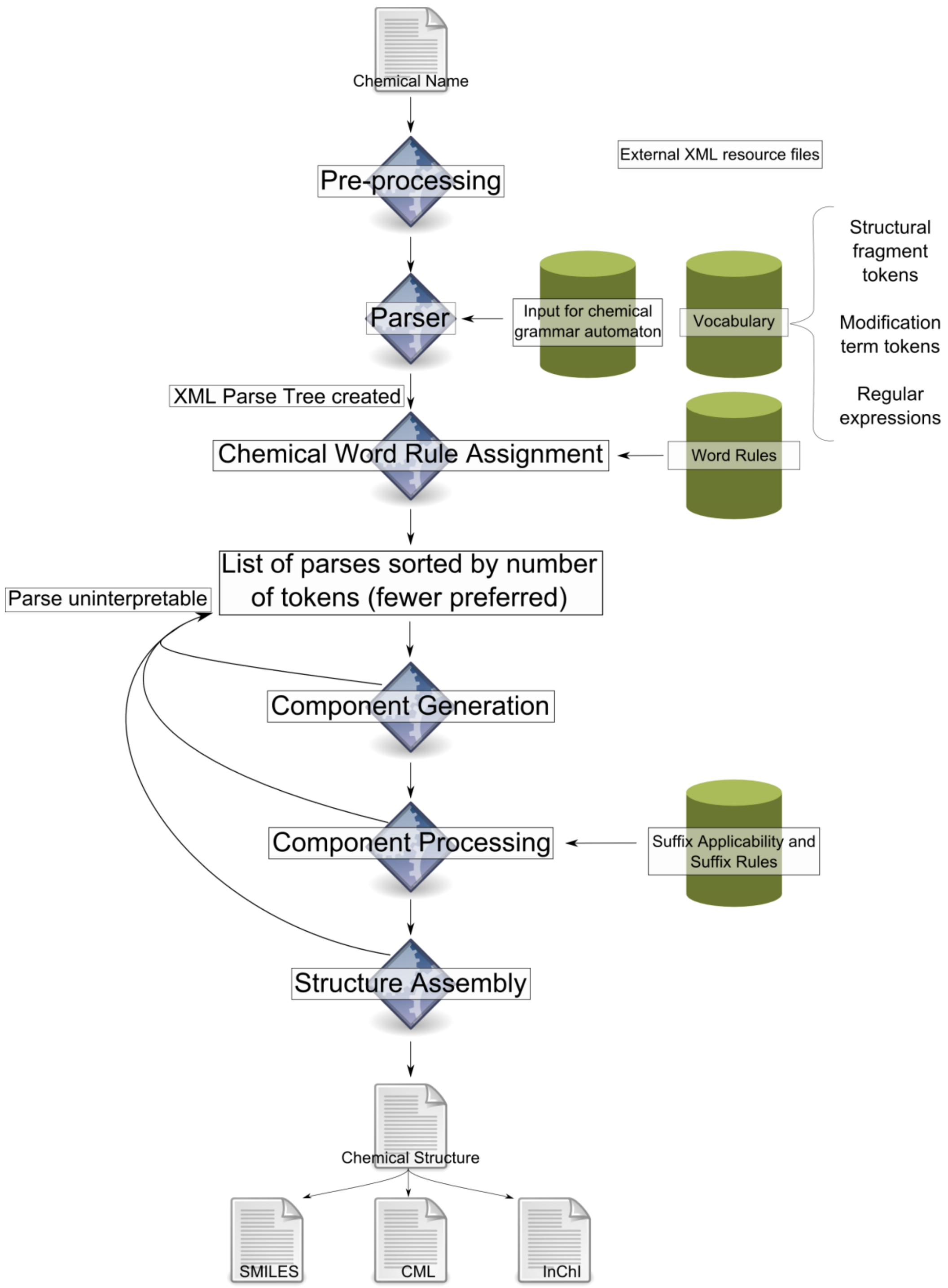


Figure 3-5 Components of OPSIN's architecture, showing the process from chemical name through to a structure

3.2.3 Pre-processing

To simplify the recognition of terms by the parser a normalisation step based on simple string manipulation was incorporated. This manipulation is used to normalise the majority of representations for Greek characters, primes and other miscellaneous symbols. Additionally the traditional British spelling of sulfur is normalised to remove the requirement of having two lexical variants for all terms incorporating the substring ‘sulf’. After normalisation all characters in the chemical name are printable ASCII characters. This property is utilised as a speed optimisation during tokenisation (Section 3.2.4.3). Examples of the result of this string normalisation can be seen in Table 3-1.

Input string	Normalised output string
λ	lambda
lambda	lambda
.lambda.	lambda
$l	lambda
sulphuric acid	sulfuric acid
`	'
′	'
“	''
⁗	''''
±	+-
ᴅ	D
æ	ae
é	e

Table 3-1 Example input and normalised output

3.2.4 Tokenisation and parsing

3.2.4.1 Introduction

Chemical nomenclature can be thought of as being an artificial language. However unlike most artificial languages, such as programming languages, the morphemes of chemical names are often not delimited. As a result, tokenisation of chemical names must, at least to some extent, rely on a predefined lexicon of what may be found in a chemical name. The approach taken by Corbett and Murray-Rust[60] was to create all possible tokenisations for a chemical name based on the program’s lexicon. However, as such an approach does not take into account the context, many tokenisations will ultimately be found to be incorrect. For example ‘propan-2-ol’ would be tokenised to [‘prop’, ‘an’, ‘-’, ‘2-’, ‘ol’] and [‘propa’, ‘n-’, ‘2-’, ‘ol’], for which the latter is clearly wrong (the ‘n-’ token is the same that would be found in ‘n-butane’). As all possible tokenisations must be generated, assuming the number of places where tokenisation is ambiguous is *n,* and that each instance results in two possibilities, this approach leads to 2^n possible tokenisations. In longer systematic names, this can cause an impractically large number of tokenisations to be generated and for the tokenisation processes to take an unacceptably long time.

The approach taken by Kirby et al.[90] avoided this problem by associating the morphemes with terminal symbols from their formal grammar. During parsing, identified morphemes may then be restricted to those with terminal symbols that are valid at that point in the grammar. The approach favoured the longest identified morpheme, with backtracking and selection of a shorter morpheme being performed if at a point in the chemical name no appropriate morphemes can be identified. An apparent drawback of this approach is that in cases where the grammar is ambiguous and selection of a shorter morpheme would yield an alternate parse this tokenisation would not be discovered.

The approach taken by OPSIN is similar to the approach of Kirby et al. except that all possible parses are evaluated.

3.2.4.2 Tokenisation algorithm

Before this system is explained, it should be first qualified what is actually being tokenised. Rather than attempting to write a grammar that describes all possible chemical names OPSIN's grammar instead describes a chemical "word" which in this context refers to the smallest meaningful unit of language. A chemical name relates to these words by the following grammar:

```
Chemical ::= Word+

Word ::= Substituent | Full | FunctionalTerm

Substituent ::= Token+

Full ::= Substituent* MainGroup

MainGroup ::= Token+

FunctionalTerm ::= Token+
```

Where a "substituent" word describes a fragment of a chemical compound e.g. 'ethyl', a "full" word describes a standalone chemical word e.g. 'benzene' or 'ethylbenzene' and a "functional term" describes a modification term e.g. 'ester'.

Table 3-2 gives a few examples of the result of dividing chemical names into words. 'Vitamin C' is interpreted as one word, as only when considered as a single unit does it have its intended meaning. Similarly 'acetic acid' is treated as one word as 'acid' on its own is not currently treated as being meaningful. It should be emphasised that the definition of a word is not defined by whitespace and instead the tokenisation process will determine the word boundaries.

Name	Number of Words	Word Types
Ethanoate	1	Full
Ethyl	1	Substituent
Ethylethanoate	1	Full
Ethyl ethanoate	2	Substituent, Full
Acetic acid	1	Full
Acetic anhydride	2	Full, Functional term
Vitamin C	1	Full

Table 3-2 Examples of chemical names with the number of words and types, as determined by OPSIN.

OPSIN's grammar, as of v1.2.0, describes 123 discrete classes of token. Each token class can either correspond to a list of tokens (e.g. 'benzen', 'pyridin' etc.) or, for classes that are not practical to enumerate, to a regular expression that describes all tokens of that class (e.g. an expression for a locant or for von Baeyer nomenclature). These lists of tokens and regular expressions are present in external XML resource files allowing the easy addition of new vocabulary. Individual tokens are associated in this XML with attributes containing semantic information, such as for 'pyridin' the structure of pyridine and for 'tetra' that its value is 4. Additionally, the type of element that will ultimately be created for this token when OPSIN produces its XML parse tree is indicated, e.g. "group" and "multiplier" for 'pyridin' and 'tetra' respectively.

The 123 token classes in the grammar are represented for convenience as single characters (which to avoid confusion with characters in the chemical name will be referred to as token characters) and are each associated with a short textual description of their meaning (*cf*. Appendix C). The grammar dictates which arrangements of these token characters are allowed.

The arrangements of the token characters are expressed as a large regular expression. This is then compiled into a deterministic finite-state automaton using the dk.brics.automaton package[110]. A deterministic finite-state automaton is formed of states, with each state having a set of allowed transitions. These transitions correspond to the set of token characters that the automaton may consume when in that state. Each transition leads the automaton to a new state. States that correspond to an acceptable end point are called "accept states". In OPSIN's grammar, these always correspond to a transition involving the endOfSubstituent, endOfMainGroup or endOfFunctionalGroup token character.

To make maintaining and updating this regular expression tractable, it is expressed in terms of the descriptions of the grammar token characters, with aliases used for complex expressions. For example, there is an expression called "ringGroup" that describes any single ring, any von Baeyer ring system or any trivial ring system. "ringGroup" is then used as part of the expression for ring assemblies, fused systems and certain spiro systems. In v1.2.0 this regular expression alone is 1103 characters and the complete grammar is a 251,224 character long regular expression.

As previously mentioned, OPSIN does not treat whitespace as a hard delimiter; determination of what is considered a breaking white space and a white space that is part of a token is instead determined as a result of how the name has been tokenised. Tokenisation and parsing occurs simultaneously as follows:

- From the current state in the discrete finite automaton the list of allowed transitions is checked to determine which token characters are allowable next.
- For each allowed next token character, attempt to match all corresponding tokens and regular expressions against the start of the chemical name.
- If a match is successful, the grammar token character and token is recorded and this process is repeated with the state now being the state to which the transition led.
- The process terminates when no more of the name can be tokenised. The tokenisations that include the largest part of the name and end in an accept state are returned.

This process is performed iteratively and multiple routes may be found through the automaton yielding multiple parses. A parse is only successful if in addition to the requirement of ending in an accept state the next character in the name is a white space or the end of the name (Figure 3-6).

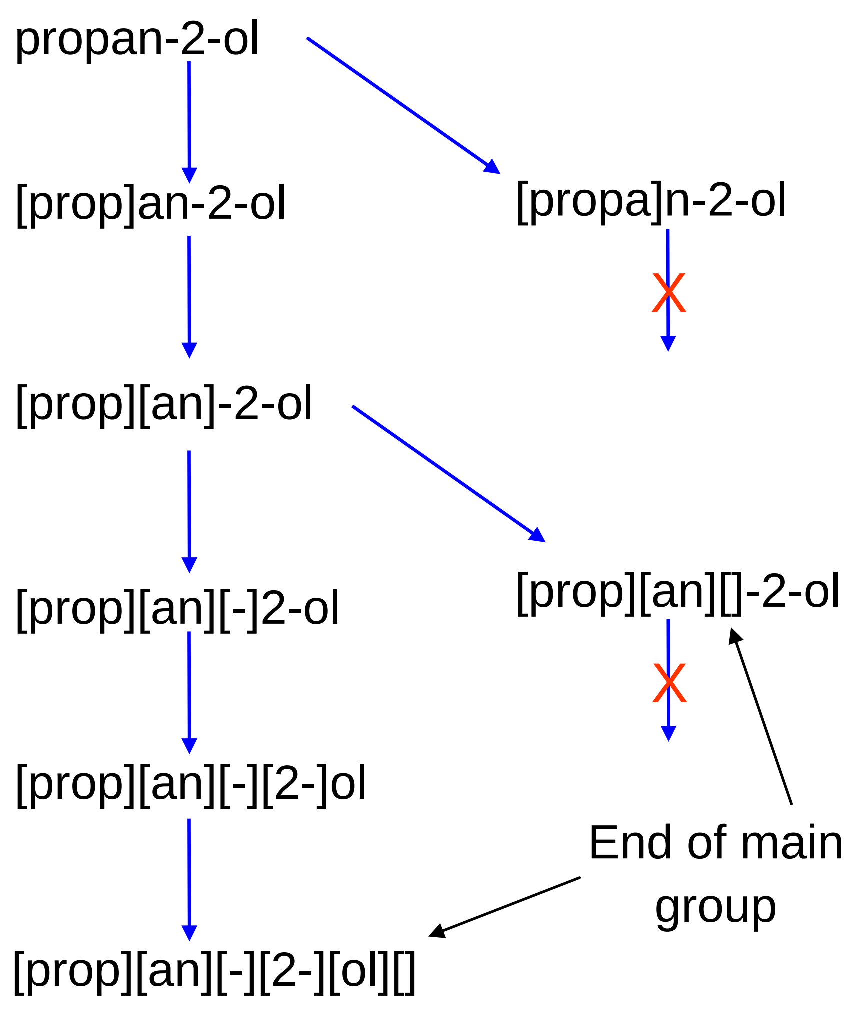


Figure 3-6 Example of how OPSIN's parser can quickly reject ungrammatical tokenisations. Note the paths through the automaton, and how only one parse reaches an acceptable end point.

3.2.4.3 Looking up tokens in the lexicon

To make the tokenisation/parsing process as fast as possible OPSIN employs a radix trie to store vocabulary tokens. A trie is a tree data structure in which strings sharing a common prefix share a common node (Figure 3-7). Each node accepts a single character and may have up to as many children as there are in the alphabet. The time taken to look up whether a string is present in a trie is practically independent of the number of lexicon entries and instead scales linearly with the length of the string that is being looked up hence yielding excellent performance even with a large lexicon.

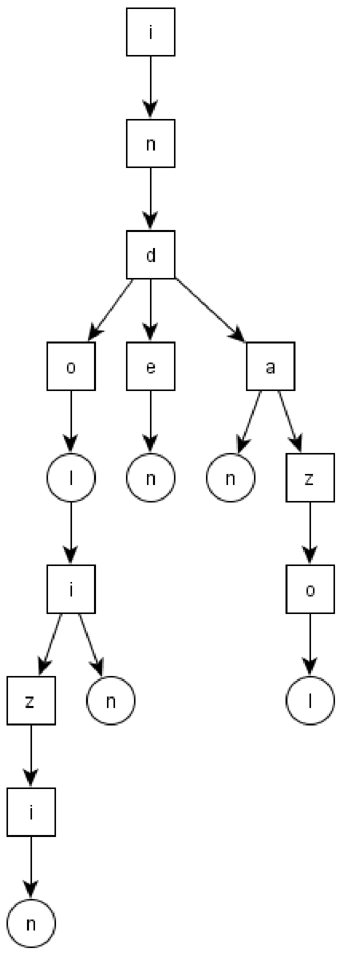


Figure 3-7 A trie describing indol, indolizin, indolin, inden, indazol and indan. Circular nodes are accepting nodes

Due to the wide lexical variety in chemical names, especially trivial names, a standard trie is memory inefficient and hence a radix trie is employed by OPSIN. In a radix trie, nodes with only a single child are merged with their child such that all non-accepting nodes have more than one child (Figure 3-8).

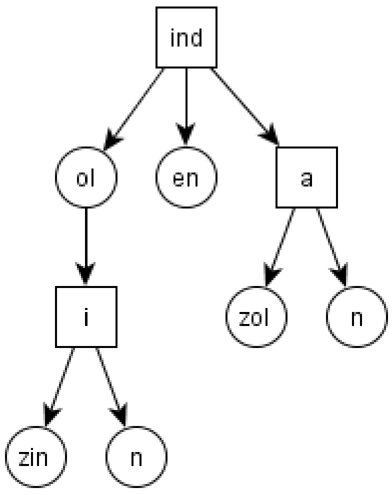


Figure 3-8 A radix trie describing indol, indolizin, indolin, inden, indazol and indan. Circular nodes are accepting nodes

A trie data structure can also be used to efficiently check for the existence of prefixes that differ by a small change such as a character insertion, deletion or transposition. Whilst not investigated in this work, extending OPSIN to suggest spelling corrections in an efficient manner is hence believed to be highly tractable.

The regular expressions that correspond to the non-enumerable token classes are compiled in advance into deterministic finite-state automata which, like the trie, have run time solely dependent on the length of string matched i.e. independent of the complexity of the regular expression the state machine describes. The process of compiling regular expressions to deterministic finite-state automata can be quite slow especially for the regular expression that describes the grammar; hence the resultant deterministic finite-state automata are serialised and only updated if the regular expressions that generate them are altered.

3.2.4.4 Generation of parses

Each word that OPSIN is able to parse will produce one or more parses; a parse being formed of a list of token/token class pairs. All possible combinations of the parses for each word are then generated. For the majority of chemical names, this process results in only one parse for the chemical name as each constituent word could be parsed unambiguously (Figure 3-9).

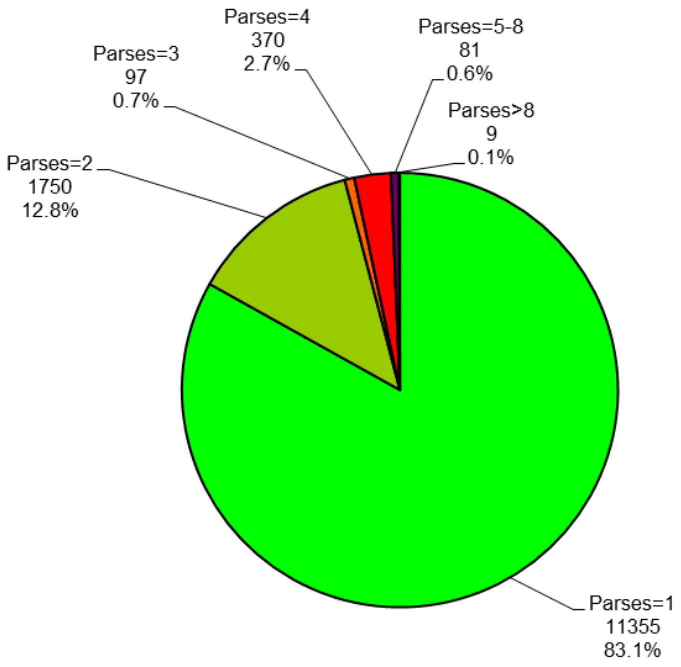


Figure 3-9 Number of parses generated from parsable IUPAC names in the December 2011 ChEBI database[5]

An example of a name in the 4 parses category was: '3,7,11,15-tetramethylhexadeca-2,6,10,14-tetraen-1-yl diphosphate' for which the following tokenisations were generated:

[3,7,11,15-, tetra, meth, yl, hexa, deca, -, 2,6,10,14-, tetra, en, -, 1-, yl] [di, phosphate]

[3,7,11,15-, tetra, meth, yl, hexa, deca, -, 2,6,10,14-, tetra, en, -, 1-, yl] [diphosphate]

[3,7,11,15-, tetra, meth, yl, hexadeca, -, 2,6,10,14-, tetra, en, -, 1-, yl] [di, phosphate]

[3,7,11,15-, tetra, meth, yl, hexadeca, -, 2,6,10,14-, tetra, en, -, 1-, yl] [diphosphate]

As the same token may appear in multiple token classes multiple parses doesn't necessarily indicate ambiguity in tokenisation.

While the path through the finite state automaton, described by OPSIN's grammar, is unambiguous for a given sequence of token characters in practice we do not know *a priori* the token classes involved or even the tokenisation. The parser instead investigates all token class/token pairs that are acceptable to the grammar and match the chemical name. Ambiguity can arise from different senses of a word, e.g. oxide can be a synonym for ether ('diethyl oxide') or mean the addition of oxygen ('trimethylphosphine oxide'). This can only be disambiguated in the next step, where the relationship between the words is considered. The other reason is that a term could exhibit ambiguity that is non-trivial to disambiguate. For example ‘tetradecyl’ is parsed as [tetradec][yl] or [tetra][dec][yl]. Cases of this type are rare and have been dealt with, on a case by case basis, as part of the Component Generation component (*cf.* Section 3.2.7.6).

3.2.4.5 Drawbacks of a regular grammar

The only significant drawback encountered in representing chemical nomenclature using a regular grammar has been the problem of representing recursive bracketing. Areas in which bracketing is not recursive, such as bracketed ring assemblies, are handled precisely by the grammar. In a regular grammar, one cannot express (to an infinite depth) a language of the form …`((a))`… where the number of open and close brackets is identical. It is, however, allowed to write the same expression in a form where the number of open and close brackets can be any number i.e. not necessarily matched. Hence, OPSIN only matches opening and closing brackets with each other after the parsing stage (Section 3.2.7.1).

3.2.4.6 Right to left parsing

By default, OPSIN parses names from left to right, but by reversing the automata that describes the grammar/non-enumerable token classes and employing tries with reversed strings, it can also be used from right to left with near identical results. Differences arise primarily from reasons outside of the parser e.g. the rightToLeft tokenisation routine currently does not remove whitespace within brackets and cannot handle the presence of extraneous words such as ‘compound with’. Genuine differences in principle may arise from the fact that the non-enumerable token class automata are greedy e.g. historically 3,4'-Bi-pyridinyl could only be parsed from right to left because 3,4'-Bi- was parsed as two locants where 4'-Bi means the atom of Bismuth that is attached to an atom with locant 4'!

The number of states in the reversed chemical grammar automaton is significantly lower, 4885 states as compared to 10747 in the left to right variant, indicating that there should be fewer routes through the automaton. This did not however translate into any improvement in tokenisation speed. The ability to parse from right to left is currently solely employed to assist in debugging which part of an unparsable name is at fault. For example, in a name such as ‘1,3-dimethyl-4-unknownyl-benzene’ OPSIN from left to right would be able to say ‘1,3-dimethyl-’ was a substituent and the name was parsable up to ‘1,3-dimethyl-4-’. From right to left, OPSIN would determine that ‘benzene’ was a full term and that the name was parsable up to ‘yl-benzene’. Hence, this combination would single out the point of failure and identify “unknown(yl)” as a potential vocabulary term.

3.2.4.7 XML generation

After parsing, an XML element is created for every token, except those that lack semantic meaning (e.g. an optional ‘e’ or an optional hyphen), to yield an XML parse tree. It should be noted that tokens from different token classes need not create different elements. For example, ‘chloro’ and ‘meth’ are tokens in different token classes but both produce a “group” element. These elements become children of `substituent`, `root` and `functionalTerm` elements with the special end of word grammar token characters being used to facilitate this chunking. These in turn are children of `word` elements. This is best illustrated with an example (Figure 3-10).

```
<molecule name="ethyl (1R,5S)-8-(chloromethyl)-8-azabicyclo[3.2.1]oct-2-ene-3-carboxylate">
  <word type="substituent" value="ethyl">
    <substituent>
      <group value="CC" labels="1/2" valType="SMILES" usableAsAJoiner="yes" type="chain" 
subType="alkaneStem">eth</group>
      <suffix value="yl" type="inline">yl</suffix>
    </substituent>
  </word>
  <word type="full" value="(1R,5S)-8-(chloromethyl)-8-azabicyclo[3.2.1]oct-2-ene-3-
carboxylate">
    <substituent>
      <stereoChemistry type="stereochemistryBracket">(1R,5S)-</stereoChemistry>
      <locant>8-</locant>
      <openbracket value="(">(</openbracket>
      <group value="-Cl" labels="none" valType="SMILES" type="substituent" 
subType="halideOrPseudoHalide">chloro</group>
    </substituent>
    <substituent>
      <group value="C" labels="1" valType="SMILES" usableAsAJoiner="yes" type="chain" 
subType="alkaneStem">meth</group>
      <suffix value="yl" type="inline">yl</suffix>
      <closebracket value=")">)</closebracket>
      <hyphen value="-">-</hyphen>
    </substituent>
    <root>
      <locant>8-</locant>
      <heteroatom value="N" valType="SMILES">aza</heteroatom>
      <multiplier value="2" type="VonBaeyer">bi</multiplier>
      <vonBaeyer>cyclo[3.2.1]</vonBaeyer>
      <group value="CCCCCCCC" labels="1/2/3/4/5/6/7/8" valType="SMILES" usableAsAJoiner="yes" 
type="chain" subType="alkaneStem" subsequentUnsemanticToken="-">oct</group>
      <locant>2-</locant>
      <unsaturator value="2" subsequentUnsemanticToken="-">ene</unsaturator>
      <locant>3-</locant>
      <suffix value="carboxylate" type="root">carboxylate</suffix>
    </root>
  </word>
</molecule>
```

Figure 3-10 XML parse tree produced for ‘ethyl (1R,5S)-8-(chloromethyl)-8-azabicyclo[3.2.1]oct-2-ene-3-carboxylate’. For this name, only one parse is produced.

3.2.5 CAS index name uninversion

CAS index names are employed by CAS to allow the parent group that denotes the most senior functionality of a molecule to be at the front of the name and hence used for indexing. This offers significant advantages for alphabetic indexing in which the same group with different substituents could end up in completely different places in the index. The process for inversion of chemical names is briefly documented in the CAS nomenclature guidelines[83]. For the simple cases the index name is simply the parent group followed by a comma, termed the inversion comma, and then the substituents each ending in a hyphen (e.g. Figure 3-11).

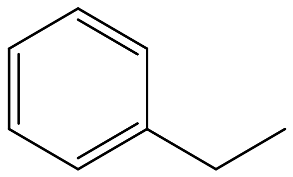


Figure 3-11 CAS name: benzene, ethyl-   IUPAC name: ethylbenzene

The inversion is somewhat more complicated when functional class nomenclature is employed (Figure 3-12), when multiplicative nomenclature is involved (Figure 3-13) or when esters are involved (Figure 3-14).

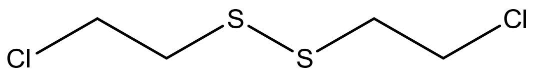


Figure 3-12 CAS name: Disulfide, bis(2-chloroethyl)   IUPAC name: Bis(2-chloroethyl) disulfide or 1,2-bis(2-chloroethyl)disulfane Note that the substituent does not have a hyphen indicating that it is not a prefix of the ‘disulfide’

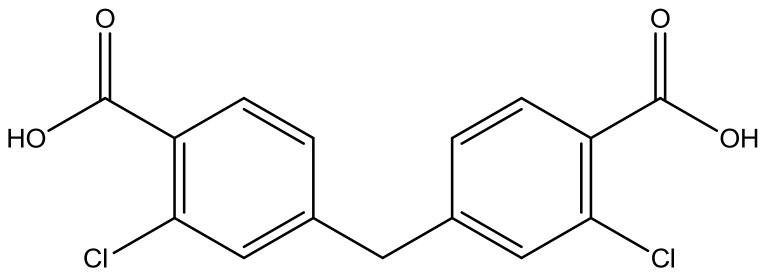


Figure 3-13 CAS name: Benzoic acid, 4,4’-methylenebis[2-chloro-   IUPAC name: 4,4'-Methylenebis[2-chlorobenzoic acid] Note that the index name has unbalanced brackets as compared to the uninverted name!

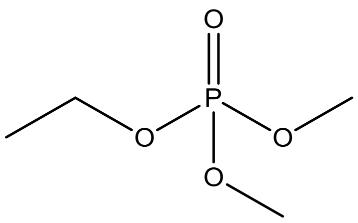


Figure 3-14 CAS name: Phosphoric acid, ethyl dimethyl ester   IUPAC name: ethyl dimethyl phosphate Note the change from phosphoric acid to phosphate

OPSIN supports CAS index names by performing an uninversion step prior to parsing. Uninversion is attempted when ‘, ’ is found in a chemical name. A comma followed by a space should be present in all well-formed CAS index names. The uninversion process involves the following steps:

- Split name on ‘, ’; the first entry in this array should be the parent group.
- Verify that if the parent group contains the space character that the words beyond the first are either ‘acid’ or something OPSIN’s parser understands.
- Iterate through the other members of the array. There should only be one but this is not enforced.
- A phrase like ‘compound with’ is ignored and a flag is set indicating subsequent words should be appended to the final name.
- The array entry under consideration is split into words by splitting on the space character.
- If a word ends with a hyphen it is a substituent. If the substituent is missing a closing bracket, a closing bracket is added to the parent group.
- If it didn’t end in a hyphen OPSIN’s parser is used to determine word type. This is used to determine how these words are added to the name. Substituents will be substituents involved in functional class nomenclature and hence go at the front of the name. Functional terms are appended to the end of the name. If the functional term is ‘ester’, the suffix of the parent group is modified. Full terms are treated in the same way as functional terms if they end in ‘ate’, ‘ite’ or are a hydrohalide. If they do not uninversion fails.
- If the word is a CAS collective index it is ignored e.g. ‘(9CI)’.
- The final name is formed from space separated substituents for functional class nomenclature then concatenated substituents and the parent group, followed by space separated functionalTerms and mixture components.

3.2.6 Chemical word rule assignment

After parsing has been completed, a chemical name will have been tokenised into substituent, full and functional term words. “Word Rules” describe the interactions between these words. For example, in 'ethyl ethanoate' (a substituent and a full word), the word rule 'ester' will be assigned indicating that the ethyl group should be connected to the charged oxygen on the ethanoate with the charge removed. Without word rules, OPSIN would not know how the ethyl fragment and ethanoate group interact. OPSIN's current word rules are listed in Table 3-3. Most word rules are only employed by chemical names using functional class nomenclature (Section 3.2.10.4).

Word Rule	**Example**
acetal	Propanal dimethyl acetal
additionCompound	Carbon tetrachloride
acidHalideOrPseudoHalide	Cyanic chloride
amide	Nitrous amide
anhydride	Acetic anhydride
biochemicalEster	Adenosine 5'-triphosphate
carbonylDerivative	Propanone oxime
divalentFunctionalGroup	Diethyl ether
ester	Ethyl ethanoate
functionalClassEster	Acetic acid ethyl ester
functionGroupAsGroup	Cyanide
glycol	Ethylene glycol
glycolEther	Ethylene glycol monomethyl ether
hydrazide	Phosphoric hydrazide
monovalentFunctionalGroup	Ethyl alcohol
multiEster	Ethyl propyl methylphosphonate
oxide	Thiophene 1,1-dioxide
polymer	Poly(ethylene)
simple	Ethylbenzene
substituent	Chloro

Table 3-3 Word rules and examples names that correspond to them

Word rule assignment is achieved by a mixture of looking at the string value of words, in particular the functional terms e.g. 'ester', and looking in more detail at the XML OPSIN has generated for a particular word. The following are two examples of word rules employed by OPSIN:

```
<wordRule name="ester" type="full">
  <word type="substituent" />
  <word type="full" endsWithRegex= "\S(at[e]?|amid[e]?|it[e]?)[\]\)\}]*"/>
</wordRule>

<wordRule name="monovalentFunctionalGroup" type="full">
  <word type="substituent" />
  <word type="functionalTerm" functionalGroupType="monoValentStandaloneGroup"/>
</wordRule>
```

Additional word rules can be added trivially by adding entries such as the above to the appropriate XML file but must be backed up by code within the program, describing the operations that the word rule requires.

When a word rule matches, the XML for the matched `word` elements are nested within a new containing `wordRule` element.

Word rules may be nested, allowing the interpretation of nested functional class nomenclature. For example, ‘choline hydrogen sulfate’ is first matched by the ester word rule and then by the biochemicalEster word rule (Figure 3-15).

```
<molecule name="choline hydrogen sulfate">
    <wordRule type="full" wordRule="biochemicalEster" value="choline hydrogen sulfate">
        <wordRule wordRule="simple" type="full" value="choline">
            <word type="full" value="choline"/>
        </wordRule>
        <wordRule type="full" wordRule="ester" value="hydrogen sulfate">
            <word type="substituent" value="hydrogen"/>
            <word type="full" value="sulfate"/>
        </wordRule>
    </wordRule>
</molecule>
```

Figure 3-15 choline hydrogen sulfate and its corresponding XML after word rule assignment. Contents of `word` elements not shown for clarity.

If the `molecule` element has multiple `wordRule` children this is indicative that the name describes either an ionic substance or a mixture. For (semi)metal halides/oxides it may be unclear as to whether it is best to represent the structure covalently or ionically. This is determined at the word rule assignment stage using cuts off on a quantitative van Arkel diagram[111]. For giant covalent structures a known limitation is that neither the ionic nor covalent form are good representations. Stoichiometry is determined by multipliers at the start of the words or specified after the name (*cf*. Section 3.2.13.1).

If no word rules match, a rule exists that allows substituents to be combined with other substituents or full words so that for example ‘ethyl benzene’ is interpreted initially as a substituent and a full word but then is converted to just one full word ‘ethyl-benzene’. At the end of word rules assignment, all words should have been assigned to a word rule otherwise an error is thrown. Whether this error is thrown for names that correspond to the substituent word rule (names formally representing radicals), e.g. ‘ethyl’, is controlled by a user-configurable switch.

3.2.7 Component generation

Component generation deals with processing nomenclature that can be efficiently acted upon without access to a connection table representation of the fragments.

3.2.7.1 XML Transformations

Some terms which are described in the grammar by regular expressions are not monolithic in nature and become more amenable if broken down further. Additionally some terms can benefit from normalisation. The XML parse tree may be manipulated to achieve this. Table 3-4 summarises these transformations/normalisations.

Term	**Example**	**How it is handled**
Superscript indication in locants	N^4 → N4	Superscript indication removed as ambiguity is not introduced
Provisional recommendation for indicating a heteroatom attached to a numeric locant	4-N → N4	Transformed into the older nomenclature for this type of locant
Greek character name in locant	ALPHA →alpha	Lower cased (OPSIN locants are case sensitive)
Added hydrogen in locant	2(9H) →2 and 9H	Added hydrogen removed from locant and added hydrogen element created
Locant that also indicates stereochemistry	1(S) → 1 and 1S	Stereochemistry removed from locant and locanted stereochemistry element created
Carbohydrate style locants	2,4,6 tri O → O2,O4,O6 tri	Transformed into more general form
Ortho/meta/para locants	o →1,ortho	Normalisation to full lower case word. Context sensitive addition of implicit '1' locant
Indicated hydrogen	1H,2H → 1H 2H	Indicated hydrogen blocks split up and locant attributes set
Stereochemistry	(1R, 2R) → 1R 2R	Converted to individual stereochemistry elements with locant attribute where locants provided
Infixes	thi oic acid → oic acid	Infixes become an attribute of the following suffix except in cases where multiplier use is ambiguous (Section 3.2.9.9a)
"Suffix prefixes"	sulfonic acid →ic acid	"suffix prefix" becomes an attribute of the following suffix
Lambda Convention	1lambda4,5 →1lambda4 4,5	Lambda Convention either assigned as an attribute of an adjacent heteroatom replacement term or formed into a new element with appropriate locant attribute

Table 3-4 Summary of XML transformations performed

As OPSIN does not employ a context-free grammar, no attempt at bracket matching is done at the grammar level. The only depth present in the XML parse tree is the division of a name into `substituent`, `root` and `functionalTerm` elements (Figure 3-16). Bracketing depth is subsequently added in by matching `openbracket` and `closebracket` elements (Figure 3-17).

The type of bracket i.e. round, curly or square is currently ignored. Any unmatched brackets will be reported and the parse will be rejected.

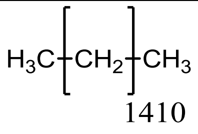
```
<substituent>
       <openbracket value="(">(</openbracket>
       <group value="-Cl" labels="none" valType="SMILES" type="substituent"
       subType="halideOrPseudoHalide">chloro</group>
</substituent>
<substituent>
       <group value="C" labels="1" valType="SMILES" usableAsAJoiner="yes"
       type="chain" subType="alkaneStem">meth</group>
       <suffix value="yl" type="inline">yl</suffix>
       <closebracket value=")">)</closebracket>
</substituent>
```

Figure 3-16 XML parse tree prior to bracket matching

```
<bracket>
       <substituent>
              <group value="-Cl" labels="none" valType="SMILES" type="substituent"
              subType="halideOrPseudoHalide">chloro</group>
       </substituent>
       <substituent>
              <group value="C" labels="1" valType="SMILES" usableAsAJoiner="yes"
              type="chain" subType="alkaneStem">meth</group>
              <suffix value="yl" type="inline">yl</suffix>
       </substituent>
</bracket>
```

Figure 3-17 XML parse tree after bracket matching

3.2.7.2 Generation of alkanes

Example	 dodectetractkiliane
General Syntax	units? tens? hundreds? thousands? unsaturation

The vast majority of alkane names are systematic in nature (Table 3-5) and hence, as the syntax for generating them is straightforward, it makes sense to generate them algorithmically rather than via enumeration. Even though only alkanes 1-4 and 11 are trivial in nature for implementation purposes it is simplest to just consider alkanes of lengths 1-9 as trivial. All other alkanes of lengths 10+ can then be considered as systematic allowing OPSIN to support creation of alkanes of length up to 9999[112]. The length of the chain may be calculated from summing the number of thousands/hundreds/tens/units in the name e.g. dodectetractkiliane is 2 + 10 + 400 + 1000 = 1412. OPSIN allows the creation of an alkane of length 11 either systematically (hendecane) or trivially (undecane). Numbering of alkanes is achieved by simply numbering the chain from one end to the other.

Alkane stem	Chain Length	Systematic?
meth (systematic = hen)	1	✗
eth (systematic = do)	2	✗
prop (systematic = tri)	3	✗
but (systematic = tetr)	4	✗
pent	5	✓
hex	6	✓
hept	7	✓
oct	8	✓
non	9	✓
dec	10	✓
undec (systematic = hendec)	11	✗
dodec	12	✓
n/a	13+	✓

Table 3-5 Alkane stems and whether they can be formed systematically

Isomers of alkanes in general are formed by systematic nomenclature but for a limited set of isomers a traditional method involving modifiers may be employed (Table 3-6). OPSIN implements these modifiers systematically by generating appropriate SMILES for the branched alkane chain. Care is taken to avoid generating nonsensical structure e.g. 'isopropane' and to respect cases where these modifiers are not used systematically e.g. 't-octyl'.

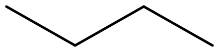
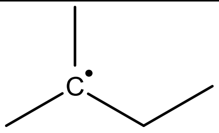
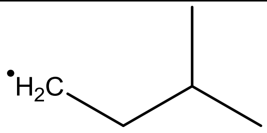
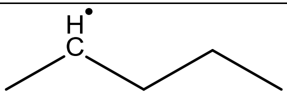
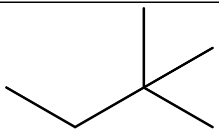
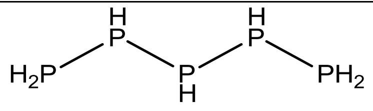
Modifier	Meaning	Example
n or normal	Straight chain (default behaviour)	 n-butane
t or tert	Atom that suffixes apply to is bonded to two methyl groups and the remaining atoms in the chain	 tert-pentyl
i or iso	The opposite end of the chain to which suffixes apply has two methyl groups attached to the penultimate atom	 isopentyl
s or sec	The second atom in the chain is used for suffixes (hence meaningless if the alkane does not have a suffix)	 sec-pentyl
neo	The opposite end of the chain to which suffixes apply has three methyl groups attached to the penultimate atom	 neohexane

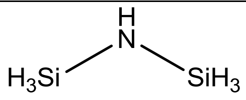
Table 3-6 Modifier prefixes for producing alkane isomers

3.2.7.3 Generation of heteroatom hydrides

Example	 pentaphosphane
General Syntax	multiplier heteroatomhydride

For chains of non-metals other than carbon and boron the chain may be named by the combination of a multiplier with the name of the hydride[77 (Rule 2.2.2)]. OPSIN implements this algorithmically to generate appropriate SMILES. Care is taken to avoid confusion between this nomenclature and cases where multiple copies of the hydride are being referred to e.g. '1,4-diazanyl-benzene'. Numbering is the same as for alkanes.

3.2.7.4 Generation of heterogeneous heteroatom hydrides

Example	 disilazane
General Syntax	multiplier heteroatom heteroatom unsaturation

If a chain is made of alternating heteroatoms it may be named by a multiplier in front of a heteroatom, where the multiplier indicates the count of that heteroatom in the chain, followed by the other heteroatom in the chain[77(Rule 2.2.3)]. OPSIN implements this algorithmically to generate appropriate SMILES. The SMILES generated depend on whether or not the chain is prefixed with 'cyclo' as this changes the composition of the chain (Figure 3-18). Numbering is the same as for alkanes.

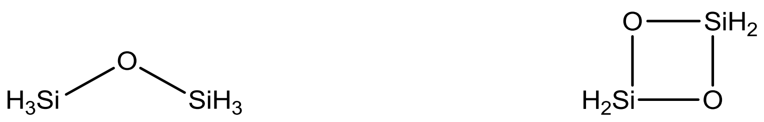
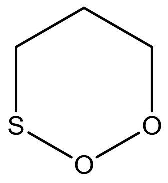


Figure 3-18 disiloxane (left) and cyclodisiloxane (right)

The nomenclature of heterogenous heteroatom hydrides overlaps with the syntax of Hantzsch-Widman nomenclature for six-membered rings (Section 3.2.9.10). The two nomenclatures are distinguishable by considering that in Hantzsch-Widman nomenclature the first heteroatom is of higher priority than the second, whilst for heterogenous heteroatom hydrides, the opposite is always true (Figure 3-19).



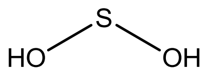
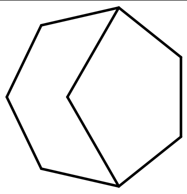


Figure 3-19 dioxathiane. Left: a HW interpretation. Right: incorrect heterogeneous heteroatom hydride interpretation

3.2.7.5 Generation of hydrocarbon ring systems

3.2.7.5a Von Baeyer nomenclature

Example	 bicyclo[3.2.1]octane
General Syntax	multiplier cyclo von Baeyer descriptor alkane

The von Baeyer system[113] is used to name polyalicyclic ring systems. This differs from fused ring nomenclature (Section 3.2.9.11) which is generally applied to systems containing at least one unsaturated ring. This distinction arises primarily from the reduction in comprehensibility when a nomenclature is applied outside of its domain rather than any difference in expressive power.

To understand von Baeyer nomenclature, first some terms need to be defined:

Bridgehead: An atom which is bonded to three or more atoms of the ring system

Bridge: A connection between two bridgeheads. This could be an unbranched chain of atoms, an atom or a bond. The latter two can be thought of as bridges of length 1 and 0 respectively.

For a system to be polycylic it must necessarily have at least two bridgeheads and three bridges. Two bridgeheads are selected and the lengths of three bridges between them form the start of the von Baeyer descriptor. If there are no further bridges then naming is complete e.g. Figure 3-20.

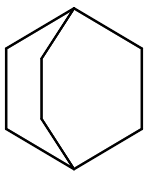


Figure 3-20 bicyclo[2.2.2]octane. This structure can be clearly seen to contain 3 bridges of length 2 between its bridgehead atoms.

Any bridges beyond the third are called secondary bridges and require locants to indicate which bridgehead atom they are between. Numbering is assigned in the order that bridges are created (Figure 3-21).

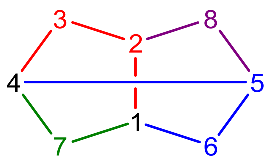


Figure 3-21 tricyclo[2.2.1.1^{2,5}]octane. The numbers correspond to the numbering the von Baeyer descriptor defines for the system. Red is the first bridge, blue is the second bridge, green is the third and purple is the fourth bridge (a secondary bridge).

The von Baeyer descriptor is almost always followed by a description of an alkane, although a heteroatom hydride (Section 3.2.7.3) is also allowed (Figure 3-22). The length of the alkane chain can, and indeed is by OPSIN, checked to assure that it is equal to the sum of the length of the bridges plus two. Additionally the multiplier preceding the von Baeyer descriptor is verified as being equal to the number of bridges plus one.

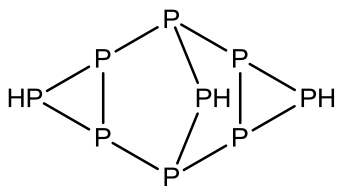


Figure 3-22 tetracyclo[3.3.1.0^{2,4}.0^{6,8}]nonaphosphane

OPSIN interprets von Baeyer nomenclature by algorithmically generating appropriate SMILES. Where superscript indication is missing OPSIN heuristically attempts to determine what is a locant and what is a bridge length indication by assuming the locant will be larger (secondary bridges are typically short in length as longer bridges are preferred for the earlier bridges in the name).

All features of von Baeyer nomenclature are supported with the exception of alternating heteroatom chains. This limitation is due to the difficulty in determining which heteroatom should be used such as to have the correct number of each heteroatom in the system with no atom being the neighbour of a heteroatom of the same element and also due to the negligible usage of this nomenclature. This difficulty can be seen in the fact that specialised nomenclature is required in some cases to actually specify which heteroatom is at locant 1! (Figure 3-23).

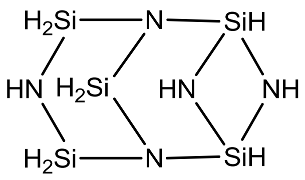
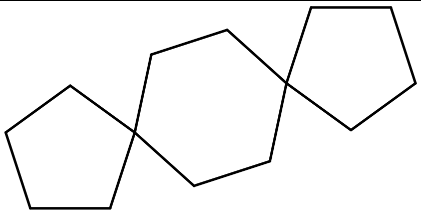


Figure 3-23 *1N*-tricyclo[3.3.1.1^{2,4}]pentasilazane (not OPSIN interpretable) or 1,3,5,7,10-pentaaza-2,4,6,8,9-pentasilatricyclo[3.3.1.1^{2,4}]decane (preferred name[78], OPSIN interpretable)

3.2.7.5b Monocyclic Spiro nomenclature

Example	 dispiro[4.2.4.2]tetradecane
General Syntax	multiplier? spiro von Baeyer descriptor alkane

A spiro fusion is one in which two rings share a single atom. Ring systems formed of monocyclic rings (i.e. not fused rings) may be named in a similar way to von Baeyer nomenclature[114(Rule SP-1)].

The von Baeyer descriptor is interpreted from left to right using a carbon atom that will become a spiro centre upon construction of the ring system. The first number in the descriptor describes the number of carbon atoms forming the link from the starting atom back to itself. Subsequent numbers describe the number of carbon atoms forming a link to a new spiro atom or back to a previous spiro atom. Algorithmically, the point at which the numbers begin describing links back to previous spiro centres may be determined by examination of the starting multiplier which indicates how many spiro centres are expected in the ring system. Numbering proceeds in the order that the links are created (Figure 3-24).

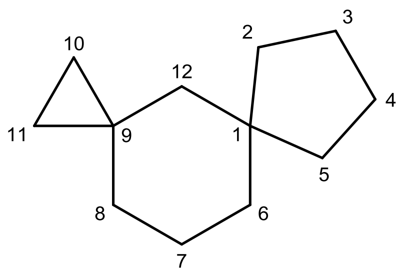


Figure 3-24 dispiro[4.3.2.1]dodecane. Atom 1 is the starting spiro atom. Atoms 2-4 are described by the '4' from the descriptor, atoms 6-8 by the '3', atoms 10-11 by the '2' and atom 12 by the '1'.

For tri and higher spiro systems it is in some cases recommended and other cases required that superscripted locants be used to indicate which spiro atom a link connects to (Figure 3-25).

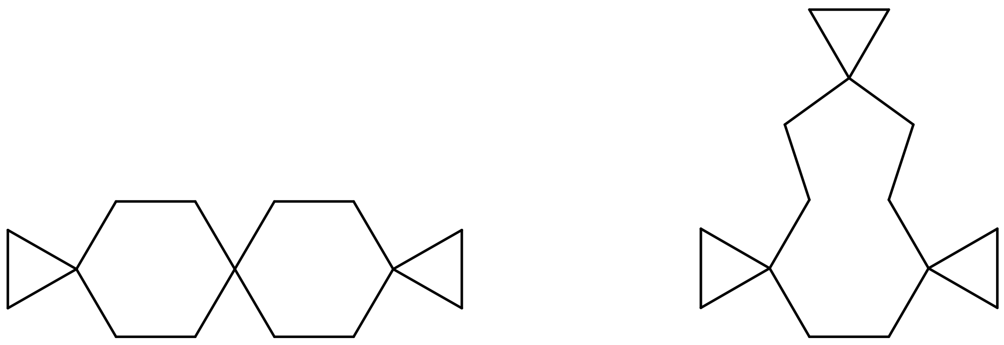


Figure 3-25 trispiro[2.2.2.2.2.2]pentadecane (left) and trispiro[2.2.2^6.2.2^{11}.2^3]pentadecane (right) showing the effect of superscripted locants

The use of superscripts is also essential if a spiro atom is visited more than twice e.g. Figure 3-26.

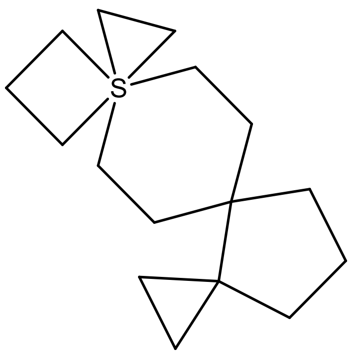


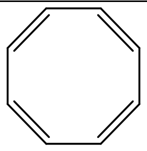
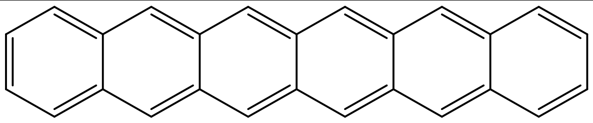
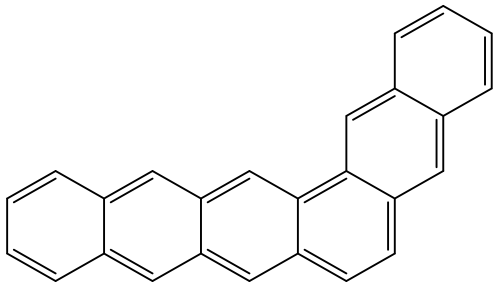
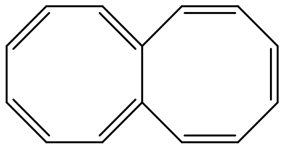
Figure 3-26 7λ^6-thiatrispiro[2.0.2.2^7.3^7.2^4.3^3]heptadecane

OPSIN interprets this nomenclature by algorithmically generating the SMILES described by the von Baeyer descriptor. A check is performed to verify that the number of atoms in the von Baeyer descriptor + the indicated number of spiro atoms as given by the multiplier is equal to the number of atoms in the alkane following the von Baeyer descriptor. Unlike in the case of von Baeyer nomenclature OPSIN requires indication that number are superscripted; the reason for this is that it is not possible to know from the name's syntax whether or not a number is expected to be followed by a superscripted number (*cf*. Figure 3-25).

OPSIN supports all rules for spiro systems formed from monocyclic rings with the exception of the generalisation to heteroatom hydrides instead of alkanes.

3.2.7.5c Other hydrocarbon ring nomenclature

The IUPAC nomenclature of fused rings[115(Rule Fr-2.1)] defines numerous micro syntaxes for naming specific types of hydrocarbon ring systems. OPSIN has complete support for all of these micro syntaxes. They are implemented by using the value of the required locant/multiplier to algorithmically generate the SMILES for the ring system (Table 3-7). All of these systems have a minimum value below which they are undefined e.g. [2]annulene is undefined as a ring of size 2 is impossible.

Example	**Nomenclature description**
 [8]annulene	Defines a ring with the maximum number of non-cumulative double bonds of size given by the bracketed number
 hexacene	A chain of *n* linearly fused benzene rings, where *n* is defined by the multiplier
 hexaphene	A chain of $(n/2) + 1$ (or $(n + 1)/2$ if *n* is odd) linearly fused benzene rings fused at 120^{o} to $(n/2) - 1$ (or $((n + 1)/2) - 1$ if *n* is odd) more linearly fused benzene rings, where *n* is defined by the multiplier
 octalene	Two rings of size *n*, with the maximum number of non-cumulative double bonds, where *n* is defined by the multiplier

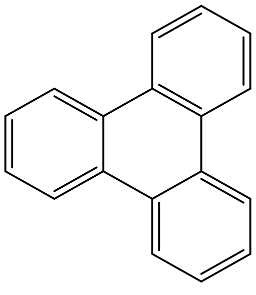
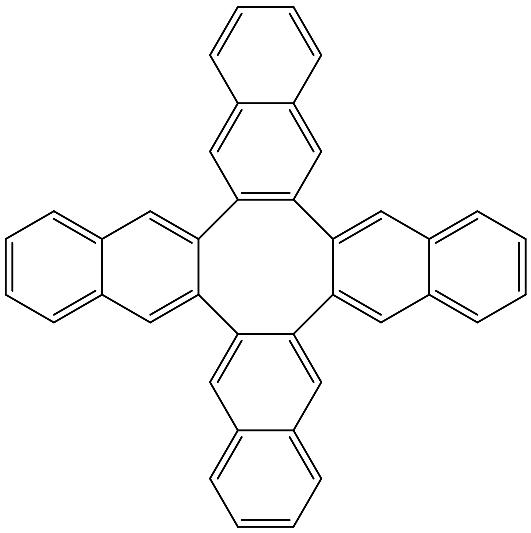
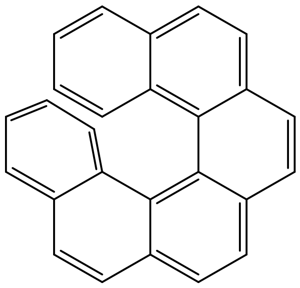
 triphenylene	*n* benzene rings fused to alternating sides of a ring of size 2n, where *n* is defined by the multiplier
 tetranaphthylene	*n* naphthalene rings 2,3-fused to alternating sides of a ring of size 2n, where *n* is defined by the multiplier
 hexahelicene	*n* benzene rings fused in a helical arrangement, where *n* is defined by the multiplier

Table 3-7 Micro syntaxes for generating hydrocarbon ring systems

3.2.7.6 Rejection of parses caused by nomenclature ambiguity

While for most names with multiple parses, all bar one will fail when performing detailed processing of the name's nomenclature, there exist some cases where multiple interpretations are plausible. Usually one of these interpretations can be readily seen to be more likely than the other, often due to one interpretation not unambiguously describing a single structure. Known cases of this type can typically be dealt with early in the name to structure process.

One example is the ambiguity between longer alkane chains and shorter multiplied alkanes (Figure 3-27). This arises due to OPSIN allowing a multiplier to apply to any group including an alkaneStem. IUPAC nomenclature recognises this ambiguity and solves it by the use of group multipliers when multiple shorter chains are desired e.g. ‘tetrakis(decyl)’. OPSIN follows these recommendations except in the case where the multiplier is immediately preceded by as many locants as the multiplier’s value in which case the multiple shorter chains interpretation is used.

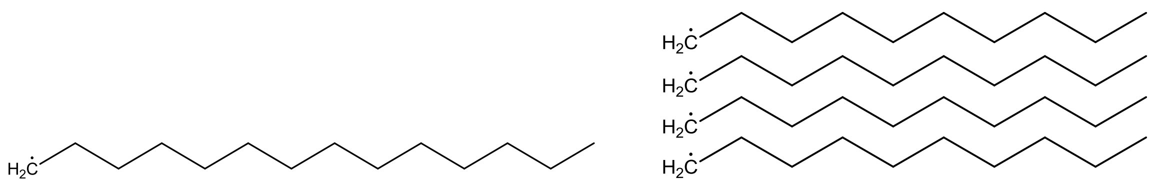


Figure 3-27 tetradecyl correct (left) and incorrect (right) interpretations

A similar, but undocumented, ambiguity occurs between multiplied phenyl rings and “polyaphene” rings (Section 3.2.7.5c). As the polyaphene interpretation is ambiguous it is not chosen unless prefixed by a locant which would locate the “yl” to a specific atom.

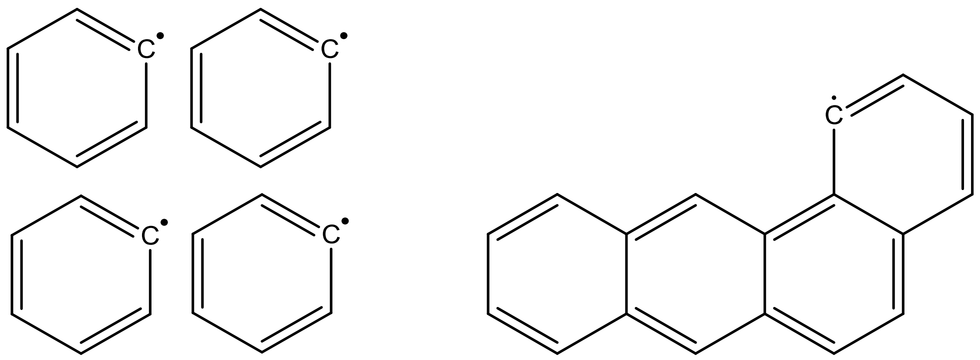


Figure 3-28 Interpretations of tetraphenyl. Left: [tetra][phenyl] Right: [tetra][phen][yl]

Figure 3-29 shows another ambiguity. In this case the phenol derivative is preferred unless the “ol” is locanted.

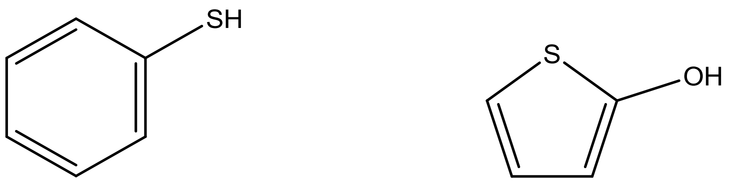


Figure 3-29 Interpretations of thiophenol. Left: [thio][phenol] Right: [thiophen][ol]

Another undocumented ambiguity occurs with heteroatom hydrides (Section 3.2.7.3) combining both the “ene” and “ium” suffix with elision of the ‘e’ on the “ene” (Figure 3-30). This clash could be avoided through the use of locants.

$Se{=}SeH^+$

Figure 3-30 Incorrect interpretation of diselenium

3.2.7.7 Handling of nomenclature irregularities

IUPAC nomenclature due to its inclusion of so many recommendations has accumulated a large number of oddities which for the most part can be dealt with prior to conversion of SMILES to structures. Handling of irregularities generally involves modification of SMILES for a group or rejection of the parse. The following are examples of aspects of nomenclature that are considered irregular to OPSIN and hence special cased.

- Presence of ‘acid’ after ‘ic’ is enforced except when followed by another word within the same word rule e.g. ‘acetic anhydride’ is allowed
- Methylenedioxy is treated as a single group and may be used to form bridges
- Multiplied ‘ethylene’s and ‘propylene’s when followed by ‘glycol’ indicate a chain interspersed with oxygens (Figure 3-31)

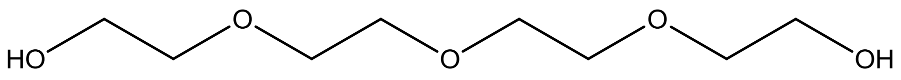


Figure 3-31 tetraethylene glycol

- ‘Xanthic acid’ and chalcogen analogues are entirely unrelated to ‘xanthene’. ‘Xanthyl’ is related to ‘xanthene’.
- Some groups have implicit locants by convention e.g. ‘anthrone’ = ‘9(10H)-anthrone’
- ‘phospho’ has a different meaning in a biochemical context to an organic chemistry context (Figure 3-32). OPSIN determines this by looking at whether the next group is an amino acid/biochemical group/carbohydrate.

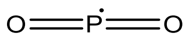
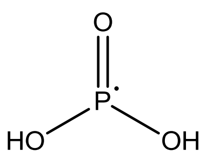



Figure 3-32 Organic interpretation of ‘phospho’ (left) and biochemical interpretation (right)

- ‘cysteic acid’ is not a synonym for ‘cysteine’
- ‘acrylamide’ is not a substituted amide ion (amide can mean [NH2-])
- If a group directly follows ‘azo’, or the like, it is implicitly multiplied (Figure 3-33)

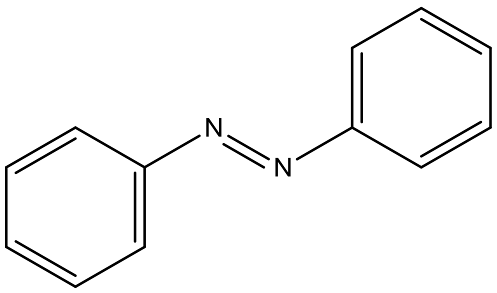


Figure 3-33 azobenzene = azodibenzene

- Acids bonded to Coenzyme A always connect via an acyl even if the name states ‘yl’ rather than ‘oyl’. Additionally even if the acid is a di-acid only one end is an acyl group (Figure 3-34).

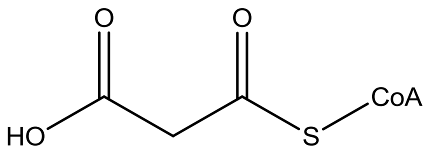


Figure 3-34 Malonyl-CoA

- ‘keto’ can be a synonym of ‘oxo’ or mean that a carbohydrate, specifically a ketose, is in the open chain form.

- fluoroantimonic acid is not a derivative of antimonic acid (Figure 3-35). Similar problems exist with some other inorganic acids.

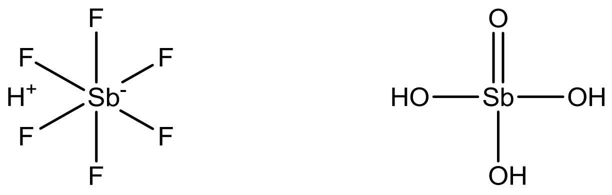


Figure 3-35 fluoroantimonic acid (left) antimonic acid (right). Systematically fluoroantimonic acid would be antimonic acid with a hydroxyl replaced by fluorine

- ‘-quinone’ is treated in the same ways as ‘-dione’ e.g. it may be prefixed with two locants.

- ‘-ylium’ may mean the removal of a hydride ion or the formation of an acylium group (Figure 3-36)

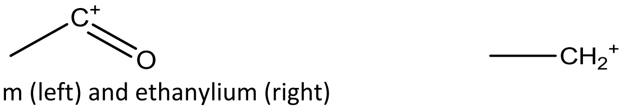


Figure 3-36 acetylium (left) and ethanylium (right)

- Multiplied phosphates may refer either to a chain of phosphates or to multiple phosphate ions (Figure 3-37). OPSIN uses the chain interpretations in preference up to a length of five phosphates.

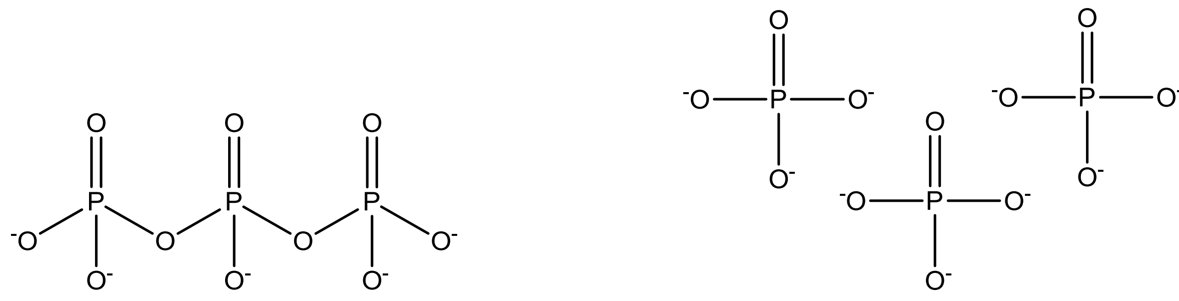


Figure 3-37 triphosphate, most likely (left); less likely (right)

3.2.8 Connection table generation

For processing more advanced nomenclature it is necessary to generate an in memory connection table representation for the fragments of the name under consideration. OPSIN achieves this using a custom SMILES reader. SMILES are read in character by character using a stack to keep track of the atom to which the next atom will be bonded. All common features of SMILES including stereochemistry are supported with some nonstandard extensions to include information not allowed in standard SMILES.

The most significant difference between normal SMILES readers and OPSIN's SMILES reader is in its interpretation of hydrogen counts. OPSIN's hydrogen model assumes that all substitutable hydrogen are implicit hence one can instead just consider the valency of an atom and from that calculate the number of hydrogens. In SMILES hydrogen may be implicit, treated as a property of an atom or treated in the same way as other atoms. The implicit case does not require explicit handling as OPSIN knows about the expected valences for organic atoms. When hydrogen atoms are treated like normal atoms, or they are the property of a non-p block metal, they are considered unsubstitutable.

In the case where hydrogen are a property of an atom OPSIN attempts to determine whether the total incoming bond order including hydrogens is consistent with the atom being in one of its standard valences. In the case that the atom is charged OPSIN attempts to understand the atom as the uncharged atom in a standard valency with a certain number of protons added or removed e.g. [NH4+] is interpreted as being in its normal valency with 1 proton added. If it is not possible to consider the atom as being in one of the expected valences a hint about the minimum final valency of the atom is set.

OPSIN supports two extensions that more naturally map to the concepts present in OPSIN's hydrogen handling model. The first is the use of 'H?' within a square bracket which is interpreted as indicating the atom has implicit hydrogen in the same way as the organic subset are interpreted. For example '[SiH?]' is interpreted to be the same as [SiH4].

The other is the ability to explicitly set the valency of an atom using the Lambda Convention (Section 3.2.9.11). This is done using the pipe character followed by the Lambda Convention valency e.g. [P|3] = [PH3]. The Lambda Convention extension is useful for specifying the valency of atoms in square brackets that when finally used will have valency higher than the sum of the intra-fragment bond orders e.g. the fragment describing 'selenoether' is [Se|2]. Describe this fragment as [Se] is incorrect as this implies 0 valency whilst [SeH2] whilst also acceptable is somewhat misleading as the fragment is really a bare selenium known to form 2 bonds. The Lambda Convention extension also allows OPSIN's valency check to be bypassed e.g. F(=O)O is rejected but [F|3](=O)O is accepted as it has been made explicit that the fluorine is expected to be that valency.

Lower case symbols in SMILES correspond to aromaticity. In OPSIN's SMILES reader it instead directly corresponds to the IUPAC's concept of maximum number of non-cumulative double bonds. This allows OPSIN to know that it may assign double bonds to atoms that cannot in their normal valency accept double bonds (Figure 3-38). OPSIN allows aromatic antimony and tellurium to allow rings with such atoms to be treated analogously to those containing arsenic and selenium.

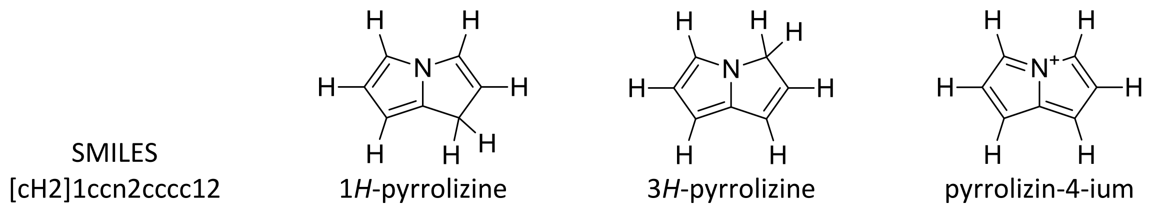
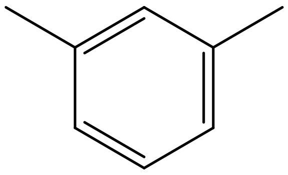


Figure 3-38 SMILES for pyrrolizine and structures that may be ultimately generated. Note that OPSIN would also accept c1ccn2cccc12 even though this is not valid SMILES.

3.2.9 Specific nomenclature handling

The majority of nomenclature manipulation occurs in the section named Component Processing in the architecture diagram (Section 3.2.2). To allow locanted operations to precede unlocanted operations, skeletal replacement nomenclature and all indications of saturation/unsaturation are handled during Structure Assembly. Note that in the cases of ring assemblies and polycyclic spiro systems (which fall under Component Processing) these pieces of nomenclature are applied prior to Structure Assembly so that the complete ring assembly or polycyclic spiro system may be assembled prior to Structure Assembly.

3.2.9.1 Groups with indeterminately positioned structural features

Example	 1,3-xylene
General Syntax	locant trivialGroup

Some trivial names do not describe a particular structure but instead multiple structures. In these cases locants preceding the trivial name may be used to specify a specific structure. In some cases such as for 'camphorsulfonic acid' (Figure 3-39) unless specified otherwise a particular locant is assumed.

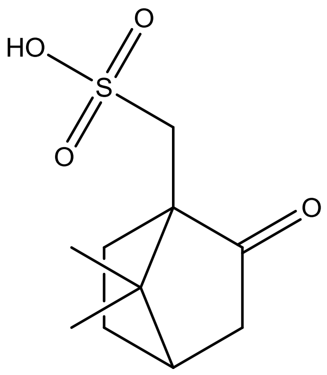
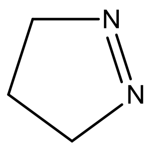


Figure 3-39 camphorsulfonic acid or more precisely 10-camphorsulfonic acid

To avoid the need to enumerate all possible combinations of locants in front of a group, OPSIN includes attributes for adding groups (*cf*. 1,3-xylene), higher order bonds (Figure 3-40) and heteroatoms (Figure 3-41) to a group



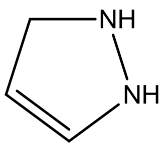
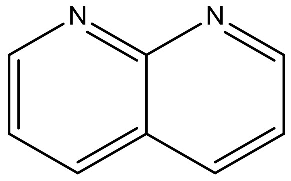


Figure 3-40 1-pyrazoline (left) and 3-pyrazoline (right)



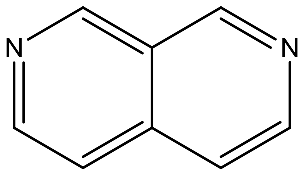


Figure 3-41 1,8-naphthyridine (left) and 2,7-naphthyridine (right)

3.2.9.2 Traditional alkane/carboxylic acid locants

Greek locants may be used instead of numbers for locants on simple alkanes (Figure 3-42) and carboxylic acids (Figure 3-43). To allow application to systematic as well as trivial groups OPSIN adds these locants algorithmically. OPSIN starts numbering from the first atom/atom at which the suffix applies. Care is taken to skip the first atom in the case where acid functionality is bonded to this atom e.g. 'ic acid' but NOT 'carboxylic acid'. Labelling proceeds along the chain as long as each atom has one unvisited carbon neighbour i.e. branches terminate labelling. Cyclic atoms are not labelled and terminate labelling. Groups with more than one acid group are not labelled.

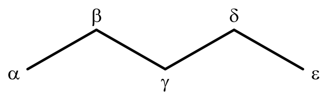


Figure 3-42 Traditional Greek locants on pentane

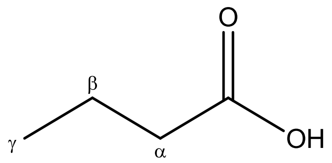


Figure 3-43 Traditional Greek locants on butyric acid/butanoic acid

3.2.9.3 Skeletal replacement nomenclature

Skeletal replacement nomenclature [76 (Rule B-4)] or "a" nomenclature refers to replacing carbon atoms in a parent structure with heteroatoms. The name "a" nomenclatures come from the fact that all the prefixes employed end with 'a' (Table 3-8). These prefixes, typically locanted, indicate which carbons should be replaced by heteroatoms e.g. Figure 3-44. OPSIN supports the full complement of "a" prefixes.

Heteroatom	"a" prefix	Plus proton	Minus hydride	Minus proton	Plus hydride
Oxygen	oxa	oxonia	oxidanylia	oxidanida	oxidanuida
Sulfur	thia	thionia	sulfanylia	sulfanida	sulfanuida
Nitrogen	aza	azonia	azanylia	azanida	azanuida
Phosphorus	phospha	phosphonia	phosphanylia	phosphanida	phosphanuida

Table 3-8 Sample "a" prefixes for skeletal heteroatom replacement and charged analogues

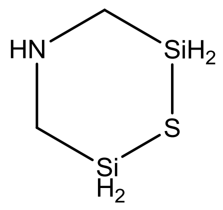
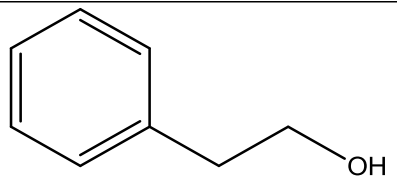


Figure 3-44 1-thia-4-aza-2,6-disilacyclohexane

This nomenclature may be used in combination with the Lambda Convention if the replacement heteroatom is not in its standard valency (*cf*. Section 3.2.9.11).

3.2.9.4 Conjunctive nomenclature

Example	 benzeneethanol
General Syntax	ring acylic group with functionality

Conjunctive nomenclature[76(Rules C-51 – C-58)] may be applied to systems formed of a cyclic component and an acyclic component containing the principle functional group. The acyclic component is numbered using Greek letters to avoid ambiguity with locants on the cyclic component (Figure 3-45). OPSIN implements conjunctive nomenclature by resolving the nomenclature that defines the ring system, resolving suffixes onto the acyclic component, renumbering the acyclic component, cloning the acyclic component if necessary (Figure 3-46) and then merging the fragments using appropriate locants or heuristically. To support [76(Rules C-812.3)] which states that radicals terminated by 'amine' may be used in conjunctive nomenclature 'ylamine' is an allowed suffix in the grammar for conjunctive nomenclature (Figure 3-47).

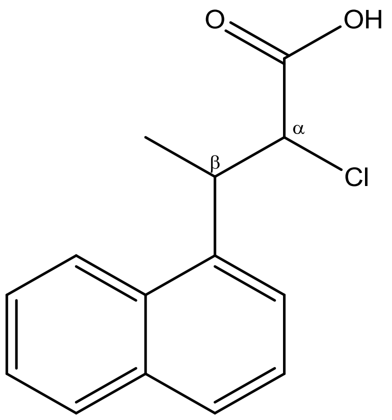


Figure 3-45 α-chloro-β-methyl-1-naphthalenepropionic acid

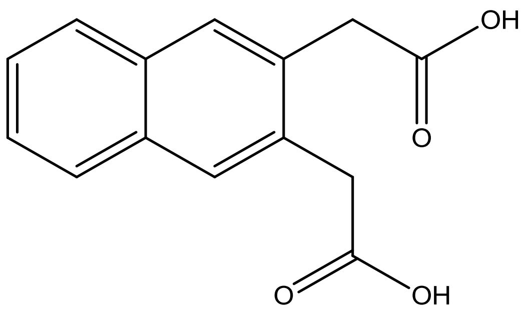


Figure 3-46 2,3-naphthalenediacetic acid

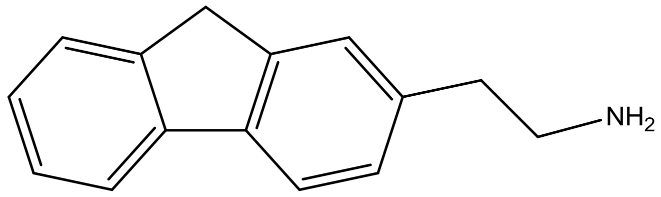


Figure 3-47 fluorene-2-ethylamine. This is handled in the same way as 'fluorene-2-ethanamine' by giving 'ylamine' the same meaning as 'amine' in this context.

3.2.9.5 Suffix handling

In IUPAC nomenclature suffixes are used to describe the principal functional group, to indicate addition/removal of charge and to indicate the presence of radicals.

OPSIN categorises suffixes into three types: normal suffixes, radical adding suffixes and charge modification suffixes. Further subdivisions are made to discriminate which are allowed to be preceded by “suffix prefixes” and/or infixes. Modification of suffixes comes under functional replacement (Section 3.2.9.9).

In general, the grammar is only specific enough to say which types of suffix are allowed on a group but is incapable of saying specifically which suffixes are allowed. This is insufficiently specific, not all suffixes are valid on all groups (Figure 3-48) and the same suffix may have different meanings on different groups (Figure 3-49).

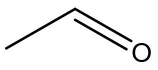


Figure 3-48 An incorrect interpretation of ‘acetal’ (correct name acetaldehyde)

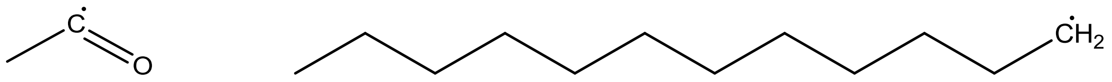


Figure 3-49 ‘yl’ has different meanings on different acidStems. Acetyl (left) and lauryl (right)

To define the effects of suffixes and to enforce rules about which groups suffixes may apply to, OPSIN uses external rules files (Figure 3-50). Adding more suffixes can be done by modifying these files but this is seldom required as the number of base suffixes in IUPAC names is finite with the vast number of possible suffixes coming from the use of infixes which OPSIN implements algorithmically (Section 3.2.9.9).

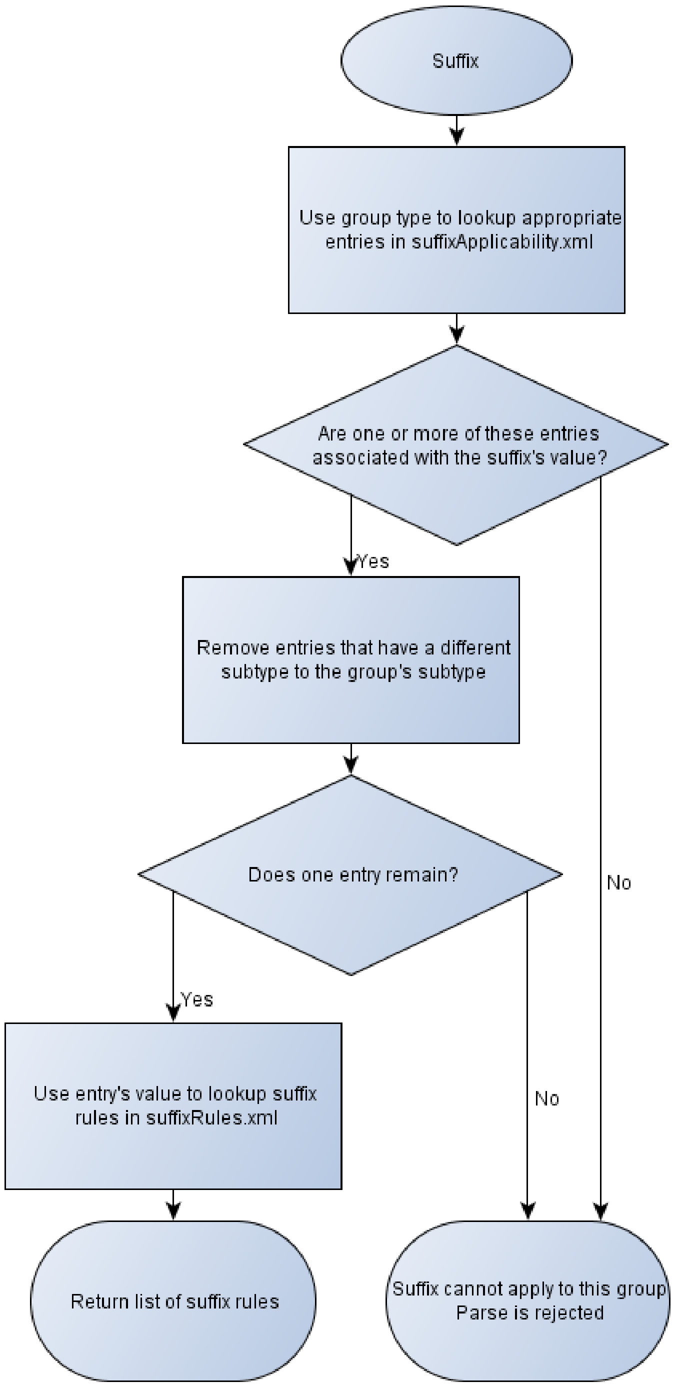

Suffix
Use group type to lookup appropriate entries in suffixApplicability.xml
Are one or more of these entries associated with the suffix's value?
Yes
No
Remove entries that have a different subtype to the group's subtype
Does one entry remain?
Yes
No
Use entry's value to lookup suffix rules in suffixRules.xml
Return list of suffix rules
Suffix cannot apply to this group Parse is rejected

Figure 3-50 Flowchart for handling suffixes. The group in question is the group to which the suffix will apply.

OPSIN defines suffixes through the use of one or more of the rules outlined in Table 3-9.

Suffix Rule	**Description**
addgroup	Adds a group defined by SMILES. Optionally the groups may be labelled, have radicals indicated or have “functional atoms” indicated
addSuffixPrefixIfNonePresentAndCyclic	Typically used to add a carbon atom before a suffix e.g. ‘pyrazinoic acid’ is interpreted the same as ‘pyrazincarboxylic acid’ would be
setOutAtom	Sets an atom to be a radical. The valency may also be specified
changecharge	Used to specify the change in charge and number of protons added/removed
addFunctionalAtomsToHydroxyGroups	Makes all hydroxyl oxygens functional
chargeHydroxyGroups	Makes all hydroxyl oxygens negatively charged
removeOneDoubleBondedOxygen	Removes a double bonded oxygen
convertHydroxyGroupsToOutAtoms	For each hydroxyl group removed a radical is added
convertHydroxyGroupsToPositiveCharge	For each hydroxyl group removed the charge is increase by one

Table 3-9 Rules used to define the effects of suffixes

OPSIN handles most suffixes as being the addition of a small group to a parent group but for inorganic acids OPSIN instead uses the suffix to mutate the structure of the acid (Figure 3-51).

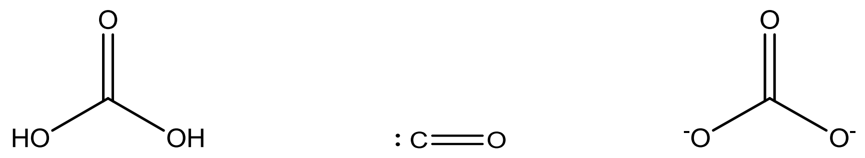
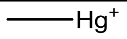


Figure 3-51 carbon**ic acid** (left), carbon**yl** (middle) and carbon**ate** (right). ‘yl’ and ‘ate’ employ the *convertHydroxyGroupsToOutAtoms* and *chargeHydroxyGroups* rules respectively

3.2.9.6 Charge and oxidation numbers

Example	 methylmercury(1+) or methylmercury(II)
General Syntax	element (chargeSpecification | oxidationNumber)

Elements, especially inorganic elements, may have their charge specified by a bracketed signed number. This can be trivially interpreted by setting the appropriate charge on the referenced atom.

Oxidation numbers are indicated using a bracketed roman number and indicate the charge that the atom would have if all its ligands were removed along with the electron pairs that were shared with the atom. OPSIN does not in general support inorganic coordinate nomenclature so attempts have only been made to make sure oxidation numbers are interpreted correctly when used in conjunction with ligands that share names with prefixes used in organic nomenclature.

Generally neutral and cationic ligands are simply the name of the ligand. As there is no suffix present to indicate that the group is a substituent, OPSIN cannot currently support these names. Hence OPSIN is allowed to make the assumption that all substituents are negative ligands with the exception of 'carbonyl' and 'nitrosyl' (Figure 3-52).

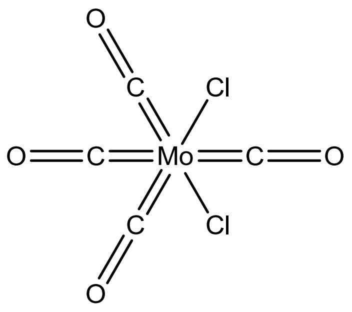
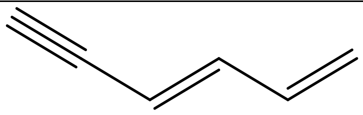


Figure 3-52 dichlorotetracarbonylmolybdenum(II). The two chloro ligands formally donate 1- charge to the molybdenum which would be 2+ with no ligands making the compound overall neutral

3.2.9.7 Indication of saturation and unsaturation

3.2.9.7a Unsaturation terms

Example	 hexa-1,3-dien-5-yne
General Syntax	alkaneStem (locant? multiplier? unsaturator)+

Unsaturation on hydrocarbons[76(Rule A-3)] is indicated using 'en(e)' and 'yn(e)'. Grammatically 'an(e)' is similar but distinct as it may not be locanted and adds no information e.g. 'ethyl' = 'ethanyl'. Unsaturation of natural products ending in 'an', 'ane' or 'anine' is also indicated in this way (Figure 3-53).

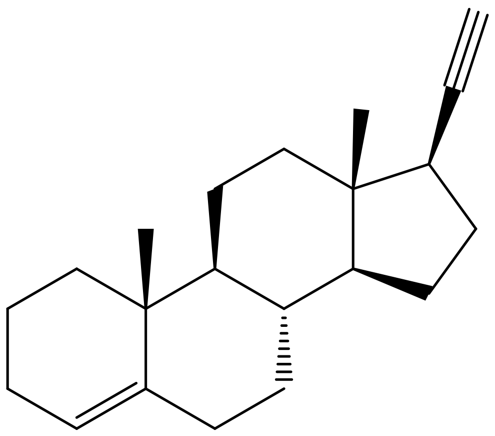


Figure 3-53 Pregn-4-en-20-yne

Where all single bonds are not equivalent, a locant is used to specify one atom in the bond to be unsaturated. The atom at the other end of the bond is implicitly the one with the locant that is 1 higher. If this is not the case, a compound locant may be used to explicitly specify the atom at the other end of the bond (Figure 3-54). OPSIN treats compound locants as if they were normal locants until the point where unsaturation is applied, at which point such locants are inspected to determine if they have a bracketed section.

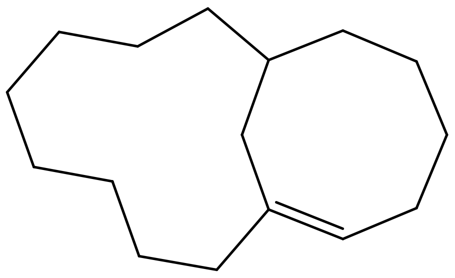
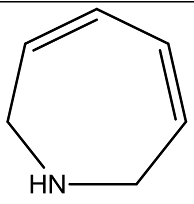


Figure 3-54 bicyclo[8.5.1]hexadec-1(15)-ene. The double bond goes between the atoms with locants 1 and 15

3.2.9.7b Hydro, dehydro, indicated hydrogen and added hydrogen

Example	 2,7-dihydro-1*H*-azepine
General Syntax	(locant? multiplier hydro)* (indicatedHydrogen)? ringSystem

Cyclic compounds are saturated and unsaturated using the prefixes hydro and dehydro respectively. As these prefixes refer to the atoms of a bond they should be in multiples of two.

OPSIN implements hydro not by actually creating double bonds but by unsetting the flag indicating that the atom may be involved in a conjugated π-system. Dehydro conversely sets the flag. These flags are used when performing kekulisation on the ring system (*cf*. Section 3.2.11). When dehydro is applied to atoms that already have the flag set an explicit triple bond is indicated (Figure 3-55). Care is taken when applying unlocanted hydro prefixes to take into account which atoms will implicitly have their flag unset due to having insufficient valency for a double bond due to the addition of a suffix or substituent. Hydro prefixes are in preference used on atoms that would otherwise be capable of supporting double bonds.

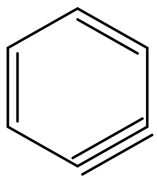


Figure 3-55 1,2-didehydrobenzene (trivial name: benzyne)

Indicated hydrogen atoms are used to indicate an atom in a ring system not involved in a double bond (Figure 3-56). Added hydrogen atoms are used to indicate the addition of a hydrogen to an atom in a ring system as a result of the addition of a suffix (Figure 3-57). Both indicated and added hydrogen are implemented by unsetting the aforementioned flag.

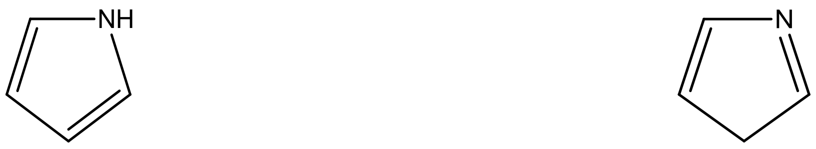


Figure 3-56 1*H*-pyrrole (left) and 3*H*-pyrrole (right)

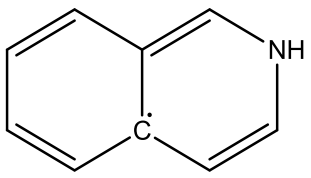


Figure 3-57 isoquinolin-4a(2H)-yl

‘Perhydro’ has historically[76(Rule A-23.1)] been used to indicate that all atoms in a ring system are unsaturated (Figure 3-58).

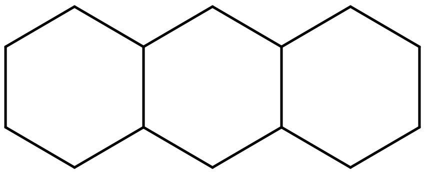


Figure 3-58 perhydroanthracene

3.2.9.8 Subtractive nomenclature

Subtractive nomenclature is used to indicate the removal of a group. It is comparatively rare in organic nomenclature and hence OPSIN only supports the most common usage; the removal of a hydroxyl using 'deoxy' (Figure 3-59). Addition of other single atom subtractive terms such as 'desmethyl' can be done at the vocabulary level but is likely to be of limited benefit due to such terms most often being used to modify trivial names e.g. drug names, that are absent from OPSIN's vocabulary.

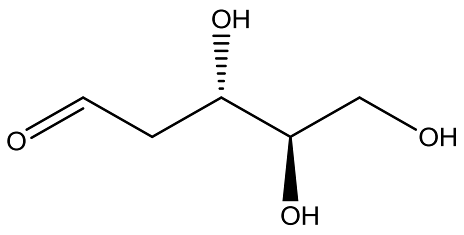


Figure 3-59 2-deoxy-D-ribose, an example of subtractive nomenclature

OPSIN implements subtractive nomenclature by normalising subtractive terms to be non-detachable prefixes (an assumption is made that such terms are likely to apply to a biochemical/carbohydrate fragment) then applying the term to the adjacent group. Care must be taken when removing atoms at chiral centres; if the centre is subsequently substituted this can be thought of as replacing the atom by a hydrogen atom which in turn is replaced hence preserving stereochemistry (Figure 3-60). As OPSIN has implicit hydrogens this is achieved by inserting a reference to a dummy "deoxyHydrogen" which will be replaced by a reference to either a hydrogen atom or a substituent atom when hydrogens are made explicit. At this point it can be determined whether the centre still is a stereocentre (*cf*. Section 3.2.14).

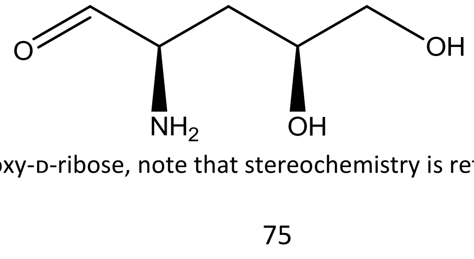
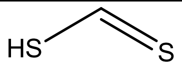


Figure 3-60 2-amino-2-deoxy-D-ribose, note that stereochemistry is retained at position 2

3.2.9.9 Functional replacement

Functional replacement involves the replacement of oxygen atoms/hydroxyl groups with other atoms or groups[77(Rule 3.4 and Table 8)]. This replacement may be indicated by the use of either prefixes or infixes with more recent recommendations tending to encourage infixes due to reduced possibilities for ambiguities. OPSIN treats functional replacements as far as possible as systematic operations. For example, OPSIN does not have 'thiol' in its list of suffixes as this can be systematically derived by combination of 'thi' with the suffix 'ol'.

3.2.9.9a Infix Functional Replacement

Example	 methanedithioic acid
General Syntax	(multiplier? infix o?)+ suffix

Infix replacement replaces oxygen atoms/hydroxyl groups within a following suffix. OPSIN implements all IUPAC infixes with the exception of chalcogen analogues of peroxo involving two different chalcogens as these require currently unsupported nomenclature to be used unambiguously.

Infix	Transformation
amid(o)	-O → N
azid(o)	-O → N=[N+]=[N-]
bromid(o)	-O → Br
chlorid(o)	-O → Cl
cyanatid(o)	-O → OC#N
cyanid(o)	-O → C#N
dithioperox(o)	-O- → SS
diselenoperox(o)	-O- → [Se][Se]
ditelluroperox(o)	-O- → [Te][Te]
fluorid(o)	-O → F
hydrazid(o)	-O → NN
hydrazon(o)	=O → =NN
imid(o)	=O → =N
iodid(o)	-O → I
isocyanatid(o)	-O → N=C=O
isocyanid(o)	-O → [N+]#[C-]
isothiocyanatid(o)	-O → N=C=S
isoselenocyanatid(o)	-O → N=C=[Se]
isotellurocyanatid(o)	-O → N=C=[Te]
nitrid(o)	=O and -O → #N
perox(o)	-O- → OO
selen(o)	=O or -O → =[Se] or -[SeH]
tellur(o)	=O or -O → =[Te] or -[TeH]
thi(o)	=O or -O → =S or -[SH]
thiocyanatid(o)	-O → SC#N
selenocyanatid(o)	-O → [Se]C#N
tellurocyanatid(o)	-O → [Te]C#N
hydroxim(o)	=O → =NO

Table 3-10 OPSIN supported infixes and the transformations they describe. Hydroxim is not an IUPAC endorsed infix.

A multiplied infix may be formally ambiguous if no brackets are used to clarify whether the infix is multiplied or the infixed suffix is multiplied (Figure 3-61). OPSIN disambiguates by inspection of multiplier type e.g. bis implies multiplication of the infixed suffix and by examining the number of available oxygen (Figure 3-62)

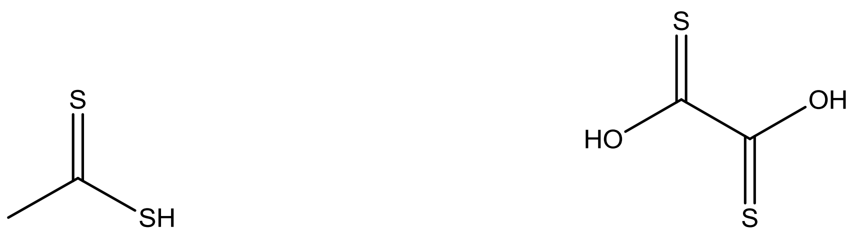


Figure 3-61 ethanedithioic acid (left, OPSIN interpretation). Incorrect interpretation (right)

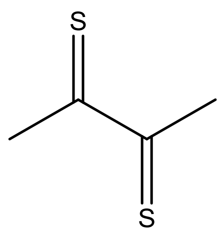


Figure 3-62 butandithione. The name clearly indicates two thione suffixes as the 'one' suffix only describes one oxygen atom.

Due to some infixes accepting more than one bond order to an oxygen, these must be acted on last to avoid problems with more specific infixes failing to apply (Figure 3-63).

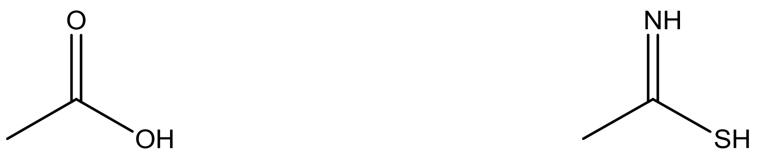


Figure 3-63 ethanoic acid (left) ethanthioimidic acid (right). The thio could apply to either oxygen whilst the imid may only apply to the double bonded oxygen

If the atom to which infix replacement applies is ambiguous this ambiguity needs to be recorded as it may be resolvable later (Figure 3-64).

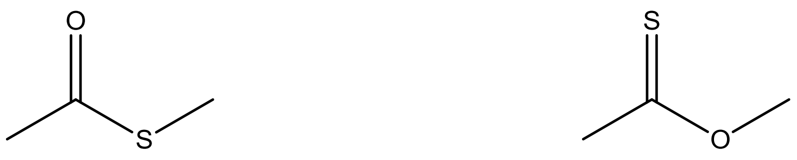
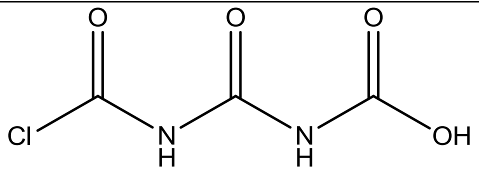


Figure 3-64 S-methyl ethanthioate (left) and O-methyl ethanthioate (right)

3.2.9.9b Prefix Functional Replacement

Example	 1-chloro-2,4-diimidotricarbonic acid
General Syntax	(locant? multiplier? prefix)+ group

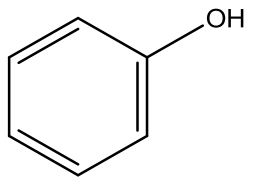
The prefixes in Table 3-11 may be employed for functional replacement. It can be quickly seen that many are identical to those employed as substituents in substitutive nomenclature hence to as far as possible avoid ambiguity, prefix functional replacement is typically only recommended for certain non-carboxylic acids.

Prefix	OPSIN classification
amido	dedicatedFunctionalReplacementPrefix
azido	halideOrPseudoHalide
bromo	halideOrPseudoHalide
chloro	halideOrPseudoHalide
cyanato	halideOrPseudoHalide
cyano	halideOrPseudoHalide
dithioperoxy	Currently Unsupported
diselenoperoxy	Currently Unsupported
ditelluroperoxy	Currently Unsupported
fluoro	halideOrPseudoHalide
hydrazido	dedicatedFunctionalReplacementPrefix
hydrazono	hydrazono
imido	dedicatedFunctionalReplacementPrefix
iodo	halideOrPseudoHalide
isocyanato	halideOrPseudoHalide
isocyano	halideOrPseudoHalide
isothiocyanato	halideOrPseudoHalide
isoselenocyanato	halideOrPseudoHalide
isotellurocyanato	halideOrPseudoHalide
nitrido	dedicatedFunctionalReplacementPrefix
peroxy	peroxy
seleno	chalcogen
telluro	chalcogen
thio	chalcogen
thiocyanato	Currently Unsupported
selenocyanato	Currently Unsupported
tellurocyanato	Currently Unsupported

Table 3-11 Prefixes for functional replacement listed by the IUPAC. Each of these prefixes corresponds to and has the same meaning as the infixes described in the previous section.

OPSIN implements this nomenclature by first classifying whether a substituent may be a functional replacement prefix and if it is classifies it as one of the following: *chalcogen*, *halideOrPseudoHalide*, *dedicatedFunctionalReplacementPrefix*, *hydrazono* or *peroxy*.

OPSIN restricts chalcogen replacement to non-carboxylic acids, the suffixes of trivial carboxylic acid stems and to aldehyde suffixes. In the case that a group has no oxygen within applicable suffixes oxygen atoms within the group may be replaced. Allowing replacement on oxygen atoms within groups allows for support of chalcogen analogues of trivial names (Figure 3-65). Explicitly adding chalcogen analogues of trivial names to OPSIN's vocabulary is generally not preferred as beside the extra effort in generating such entries, two parses will be produced: one with the chalcogen replacement prefix as a separate token and one in which it is part of the trivial name. For chalcogen analogues of rings used as components in fused ring nomenclature including the chalcogen analogue in the program's vocabulary is necessary as the grammar does not allow a prefix to precede a ring within a fused ring system.



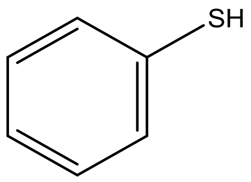


Figure 3-65 Phenol (left) and thiophenol (right). Note that 'thiophenol' is not in OPSIN's vocabulary

As with infix replacement, chalcogen replacement may be ambiguous and this ambiguity is noted as it may be resolvable later.

Peroxy replacement is treated in the same way as chalcogen replacement except that only functional oxygen atoms and etheric oxygens are considered. OPSIN prefers etheric oxygen atoms to functional oxygen atoms allowing the intended interpretation for names like peroxydicarbonic acid to be generated (Figure 3-66).

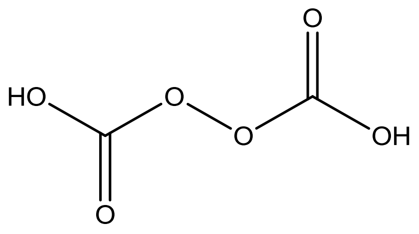


Figure 3-66 peroxydicarbonic acid

OPSIN only supports the use of *dedicatedFunctionalReplacementPrefix*es on non-carboxylic acids and enforces that they must be used for functional replacement.

Hydrazono and *halideOrPseudoHalide* functional replacement terms are also restricted to non-carboxylic acids with the additional restriction that insufficient substitutable hydrogen should be present on the atoms indicated hence precluding the substitution interpretation (Figure 3-67).

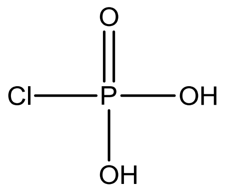


Figure 3-67 chlorophosphoric acid (functional replacement) or chlorophosphonic acid (substitution as the phosphorus in phosphonic acid has a hydrogen atom) or phosphorochloridic acid (infix functional replacement)

Care is taken when performing both infix and prefix functional replacement to have the correct charges on the modified section of the molecule and to correctly annotate which atoms are "functional atoms" (Figure 3-68).

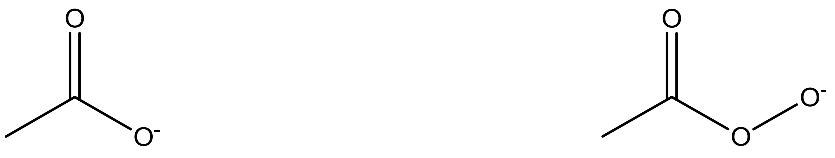
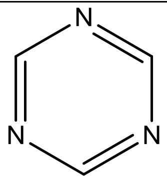


Figure 3-68 Acetate (left) and peroxyacetate (right). Note that the charge on the replacement functionality is dependent on the original functionality and that the functional atom has effectively moved.

3.2.9.10 Hantzsch-Widman nomenclature

Example	 1,3,5-triazine
General Syntax	locant? (multiplier? heteroatom)+ HWstem

Hantzsch-Widman nomenclature[116] is used to describe the structure of heteromonocycles i.e. individual rings containing at least one heteroatom. The system initially applied only to nitrogen, oxygen, sulfur and selenium but through various recommendations has been extended such that it now can be applied to all p-block elements except the noble gases. Traditionally mercury has also been included in the system although the 2004 provisional recommendations do not recommend its use[78]. A Hantzsch-Widman name is formed of one or more prefixes (Table 3-12), describing the heteroatoms in the ring, followed by a stem (Table 3-13) describing the size of the ring and whether or not it is unsaturated e.g. '1,3-oxazole'. Prefixes are preceded by locants indicating the position of the heteroatoms.

Element	Prefix*
fluorine	fluora
chlorine	chlora
bromine	broma
iodine	ioda
oxygen	oxa
sulfur	thia
selenium	selena
tellurium	tellura
nitrogen	aza
phosphorus	phospha
arsenic	arsa
antimony	stiba
bismuth	bisma
silicon	sila
germanium	germa
tin	stanna
lead	plumba
boron	bora
aluminium	aluma
gallium	galla
indium	indiga
thallium	thalla

Table 3-12 Hantzsch-Widman system prefixes *Note that the final ‘a’ is elided prior to a vowel

Ring Size	Unsaturated	Saturated
3	irene irine (nitrogen containing)	irane iridine (nitrogen containing)
4	ete	etane etidine (nitrogen containing)
5	ole	olane olidine (nitrogen containing)
6	ine/inine	ane/inane
7	epine	epane
8	ocine	ocane
9	onine	onane
10	ecine	ecane

Table 3-13 Hantzsch-Widman system stems

Note that the final ‘e’ on the stems is optional

The choice of stem for six-membered rings is dependent on the heteroatoms present in the ring and is required to avoid conflicts between Hantzsch-Widman rings and heteroatom hydrides or heteroatom chains (*cf.* Section 3.2.7.3).

OPSIN has complete support for Hantzsch-Widman nomenclature including the now deprecated support for rings with one double bond [76] (Rule B-1.2) (Figure 3-69) and deprecated heteroatom prefixes.

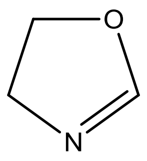


Figure 3-69 2-oxazoline; Note that the 2 refers to the position of the double bond. The position of the oxygen and nitrogen are 1,3 by widely accepted convention.

To avoid the previously mentioned ambiguity between heteroatom hydrides and Hantzsch-Widman nomenclature, OPSIN has explicit categories in its grammar for heteroatom prefixes that may be used with the 'ine' and 'ane' stems (Figure 3-70).

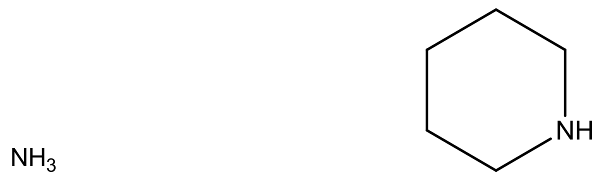


Figure 3-70 The correct interpretation of azane (left) and an incorrect Hantzsch-Widman interpretation (right). OPSIN only generates one parse for 'azane' which corresponds to the former.

The stem is used to generate a ring with appropriate saturation onto which heteroatoms are substituted. Exceptions are made to support certain ring systems having certain locants by convention e.g. 'oxazole' = '1,3-oxazole'. Additionally certain systems which will rarely mean the Hantzsch-Widman ring are blocked e.g. 'thiol' or 'seleninic acid'. The complete ring system may be subsequently used as a component in other nomenclature such as fused ring nomenclature (Section 3.2.9.11) or polycyclic spiro nomenclature (Section 3.2.9.15).

OPSIN handles the names of fused ring components recommended in FR-2.2.1(c)[115], for heteromonocycles of size greater than 10 atoms, as an extension of the Hantzsch-Widman system (Figure 3-71).

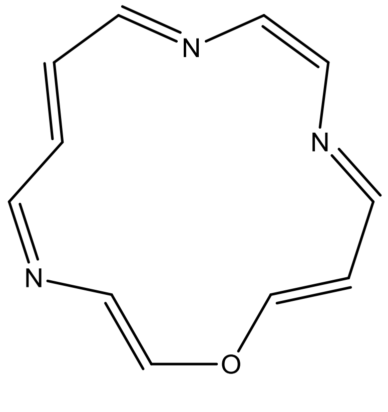


Figure 3-71 [1,4,9,12]oxatriazacyclopentadecine

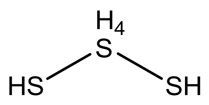
3.2.9.11 Lambda convention

IUPAC nomenclature has a concept of standard bonding number (Table 3-14) where bonding number is defined by the sum of the bond orders of all bonds to an atom. Typically an atom will have this standard bonding number hence no attempt needs to be made to specify it. If the atom is not in its standard bonding number, and does not implicitly have a non-standard bonding number (e.g. phosphorane is defined as a phosphorus atom with bonding number 5), the Lambda Convention[117] should be employed.

Standard bonding number	Elements				
3	B	Al	Ga	In	Tl
4	C	Si	Ge	Sn	Pb
3	N	P	As	Sb	Bi
2	O	S	Se	Te	Po
1	F	Cl	Br	I	At

Table 3-14 Standard bonding numbers

The Lambda Convention may be applied to heteroatoms used in skeletal replacement/Hantzsch-Widman nomenclature or directly to a group (Figure 3-72). OPSIN fully supports the Lambda Convention. In OPSIN's implementation care must be taken to distinguish between the case where the locant before the λ is needed to locate a heteroatom and cases where the locant is purely for use by the Lambda Convention.



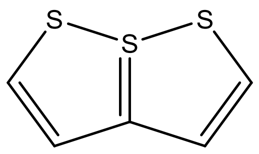
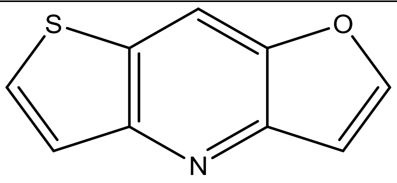


Figure 3-72 Examples of the Lambda Convention: 2λ^6-trisulfane (left) and 1,6,6aλ^4-trithiapentalene (right)

3.2.9.12 Fused Ring nomenclature

Fused ring nomenclature[115] is used to name polycyclic ring systems especially ones containing unsaturated rings. All rings in the ring system will be ortho-fused (i.e. have a bond in common) to at least one other ring.

3.2.9.12a Fused Ring System Construction

Example	 furo[3,2-*b*]thieno[2,3-*e*]pyridine
General Syntax	(fusionComponent fusionBracket)+ parentComponent

Fused ring nomenclature names are created using the names of trivially named fused ring systems, ring systems named as in Section 3.2.7.5c and individual ring names. These will hitherto be referred to as components with the rightmost component being the parent component. Atoms shared by multiple components are considered to be part of both components for the purpose of name construction (Figure 3-73). Cyclised alkanes when used as fusion components are treated as implicitly unsaturated in this nomenclature.

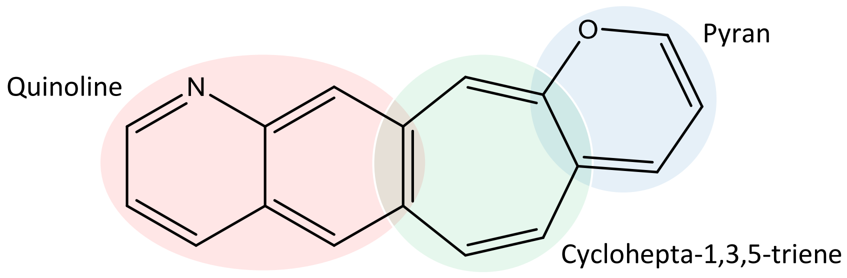


Figure 3-73 pyrano[2',3':4,5]cyclohepta[1,2-*g*]quinoline showing the components

The fusion brackets in the name describe how each component is connected to the next component. All bar references to the parent component employ numbers to refer to atoms. The parent component is instead treated as if it had no locants and instead bonds are referred to using letters (Figure 3-74). The letters typically are in the same order as the original numeric locants, except in cases where the numeric locants of the peripheral atoms are not continuous (e.g. acridine). In these cases the letters use the order the atoms would be in if the system were systematically numbered. It should be noted that purine is an exception to this.

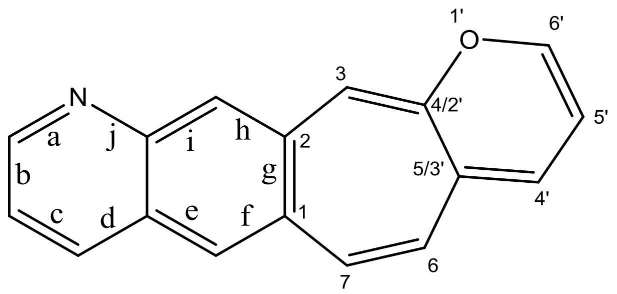


Figure 3-74 pyrano[2',3':4,5]cyclohepta[1,2-*g*]quinoline showing the internal ring numbering. This numbering is unrelated to the final numbering of the complete system

To reconstruct the fused ring system, the name is read from right to left and the components are successively fused. Components are primed, double primed etc. depending on how many components removed from the parent component they are.

Components may also be multiplied (Figure 3-75). Multiplied components are primed but may not be fused onto hence avoiding the introduction of numbering ambiguity.

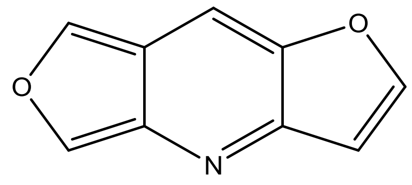


Figure 3-75 difuro[3,2-*b*:3',4'-*e*]pyridine

A fusion bracket specifies the atoms that are to be fused in each fusion (except for fusion brackets to the parent compound which specify bonds). However there are many cases in which some locants may be omitted or the entire fusion bracket omitted. In these cases OPSIN internally generates a fusion bracket with the missing locants added before proceeding as normal (e.g. Figure 3-76 and Figure 3-77).

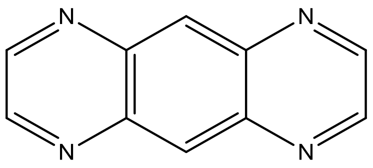


Figure 3-76 pyrazino[*g*]quinoxaline becomes pyrazino[2,3-*g*]quinoxaline

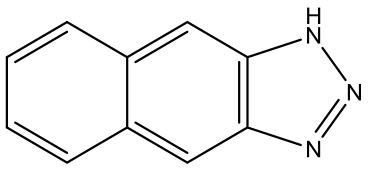


Figure 3-77 1*H*-naphtho[2,3][1,2,3]triazole becomes 1*H*-naphtho[2,3-*d*][1,2,3]triazole

Bridgehead atoms are typically omitted from fusion brackets (Figure 3-78). To get a complete list of atoms to be used in a fusion OPSIN iterates over the component's atoms from the atom indicated by the starting locant to the atom indicated by the ending locant. This list will include all atoms including bridgeheads.

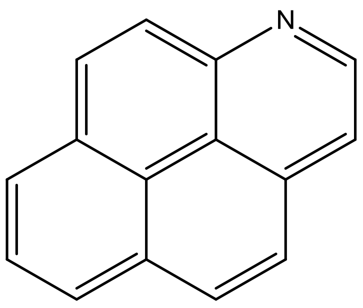
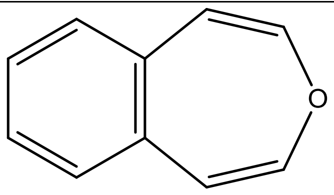


Figure 3-78 naphtho[2,1,8-*def*]quinoline (interpreted as if it were written naphtho[2,1,8a,8-*def*]quinoline)

This procedure for handling missing locants appears to be similar to that described by Matsuura[94] with the exception that OPSIN never employs numeric locants to describe atoms to use on the parent component.

With the exception of benzo fusions (Section 3.2.9.12b) and multi-parent systems (Section 3.2.9.12c), OPSIN does not support the inclusion of fused ring systems created by fused ring nomenclature as fusion components. All other aspects of fused ring system construction are believed to be supported.

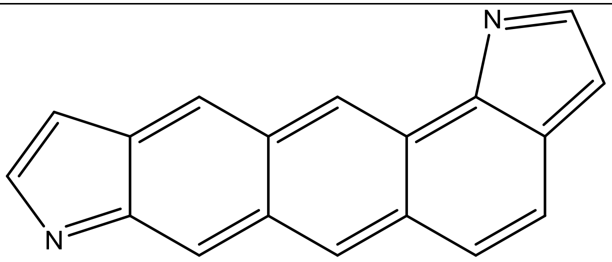
3.2.9.12b Benzo fusions

Example	 3-benzoxepine
General Syntax	locant benz(o) parentComponent

As a special case heterobicylic systems containing a benzene ring are named using a different syntax. Benzene is fused to the parent component and then the locant before the 'benzo' is used to assign the position of heteroatoms in the complete system. Although not made explicit in the nomenclature recommendations, for implementation purposes the heteroatoms must be considered to be repositioned, as their absence would mean that numbering does not necessarily start on the parent component.

Such systems may be used as fusion components and hence are processed prior to other fusion nomenclature.

3.2.9.12c Multi-parent systems

Example	 benzo[1,2-*f*:4,5-*g*']diindole
General Syntax	(fusionComponent fusionBracket multiplier)+ parentComponent

When there are multiple candidates for the parent component and all candidates are fused to the same component, this part of the ring system may be named as a multi-parent system. For these systems, a multiplier is used to indicate that a component is replicated and locants are used to indicate where these components are fused.

As long as the pairs of inter-parent components are identical, the system can also be used in cases involving more than one inter-parent component (Figure 3-79).

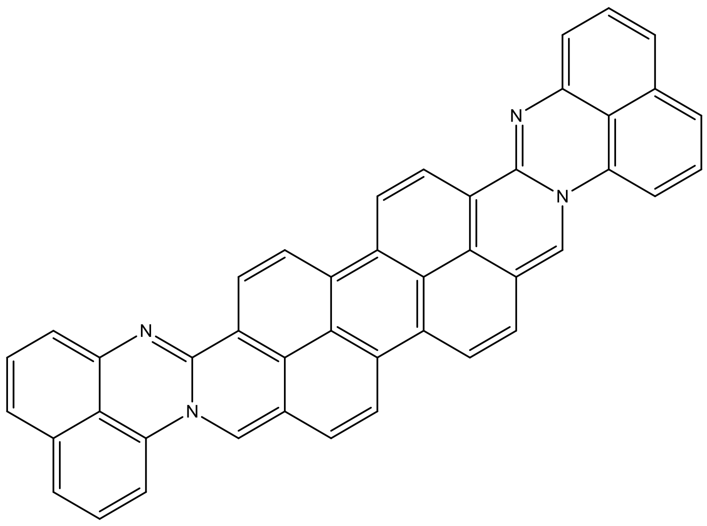


Figure 3-79 anthra[2'',1'',9'':4,5,6;6'',5'',10'':4',5',6']diisoquinolino[2,1-*a*:2',3'-*a*']diperimidine

Multi-parent systems may have further fusion performed on them and hence are processed after benzo fusions, but prior to other fusion nomenclature. The whole multi-parent system can be thought of as being the parent component for further fusion.

3.2.9.12d Idealised grid construction

Once the fused ring system has been constructed it is numbered by OPSIN. This is achieved by determining the preferred layout of the ring system on an idealised 2D grid, determining the preferred orientation and then determining the preferred peripheral numbering. The first of these steps is significantly complicated by not all sizes of rings tessellating. The system works perfectly for six-membered rings, but for all other ring sizes manipulation of ring shape or multiple orientations are possible (Figure 3-80).

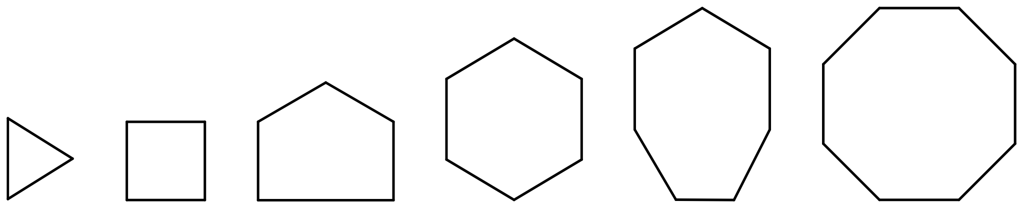


Figure 3-80 Ring shapes considered by OPSIN. Ring shapes that are recommended by Fused Ring and Bridged Fused Ring Nomenclature (1998)[115] but not by the 2004 draft recommendations[78] are not considered by OPSIN.

First, OPSIN determines the rings that comprise the fused ring system. This is achieved by calculating the smallest set of smallest rings (SSSR) from the complete ring system. These rings are then associated with their neighbouring rings.

Starting from a ring with the minimum number of neighbouring rings, ring connection tables are created. Multiple ring connection tables may be created as multiple orientations of the ring shapes may need to be considered for 5- and 7-membered rings (Figure 3-81). OPSIN considers the minimum possible number of orientations needed to enumerate all possibilities grid layouts. For example, when a ring is only involved in one fusion, only one orientation needs to be considered as the different orientations only effect the calculated position of other rings relative to the starting fusion bond. Additionally, OPSIN does not consider orientations of 5-membered rings involving fusion to the elongated bond if other orientations are possible. An example of a complete ring connection table may be seen in Figure 3-82.

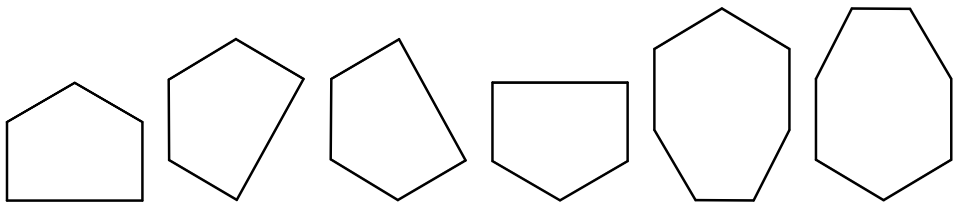


Figure 3-81 Orientations potentially considered for 5- and 7-membered rings

Ring	Ring shape	Direction	Neighbouring Ring
benzene	*standard*	1	pyridine
pyridine	*standard*	0	cyclopenta
cyclopenta	*enterFromLeftHouse*	4	pyridine
pyridine	*standard*	-3	benzene

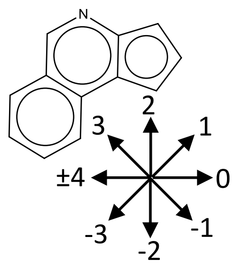


Figure 3-82 Ring connection table for cyclopenta[c]isoquinoline. The depiction shows the orientation described by this table. The depiction often, such as in this case, does not represent the final orientation of the system. Numbers are used to indicate the directions from a ring to a neighbouring ring on the idealised grid.

An example of a case where multiple orientations of a 5-membered ring need to be considered to evaluate all possible grid layouts is shown in Figure 3-83 .

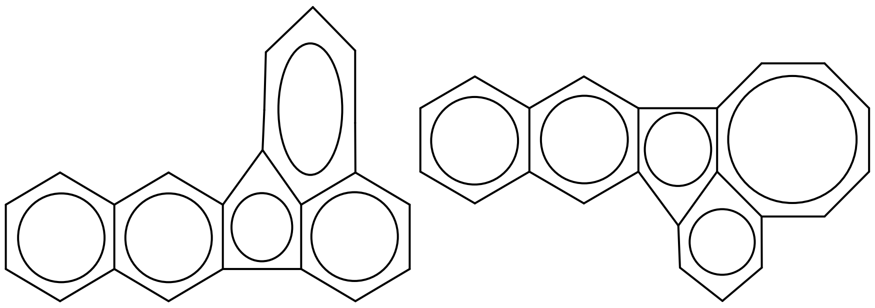


Figure 3-83 Layouts for benzo[b]cycloocta[jk]fluorine ignoring rotations and reflections. These layouts are indistinguishable until peripheral numbering is considered (right layout preferred).

Next, OPSIN attempts to eliminate those connection tables with more distorted rings. Distorted rings are recognised by the direction from one ring to another not being the opposite of the direction from the other ring back to the first ring. Generation of connection tables for ring systems that may only be drawn with distorted rings is a known limitation in OPSIN's implementation (Figure 3-84).

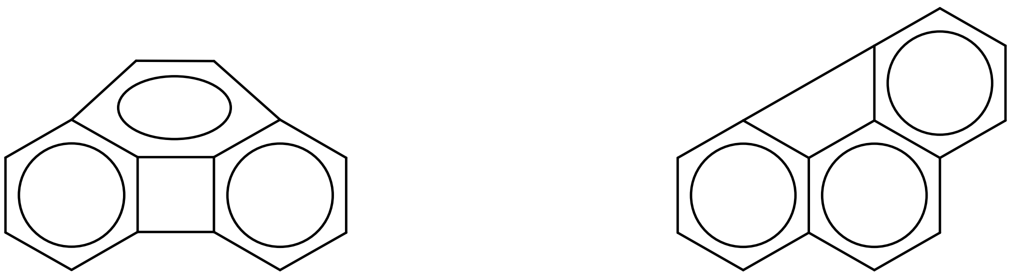


Figure 3-84 Distorted ring possibilities for cyclobuta[def]phenanthrene. OPSIN only currently considers the left one. The right one is the preferred layout for numbering.

For rings of sizes greater than 8, OPSIN supports the special case where all such rings are only fused to one other ring (Figure 3-85) but does not support the general case as this would require consideration of multiple ring orientations. As the IUPAC recommendations[115] acknowledge potential problems with the naming of systems containing such rings, and as these systems are also quite rare, this is not considered a significant limitation to OPSIN's implementation.

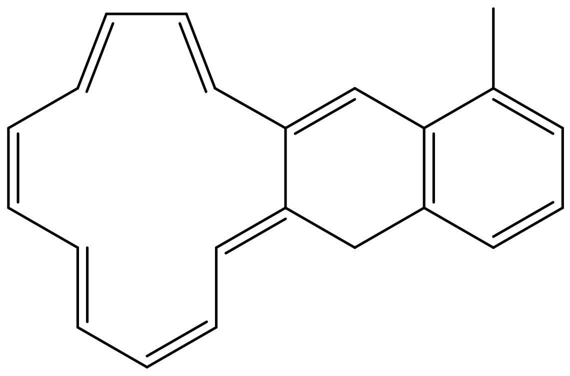


Figure 3-85 1-methyl-5H-cyclotrideca[b]naphthalene

3.2.9.12e Grid orientation

At this stage there may still be multiple ring connection tables corresponding to different grid layouts. Typically, the rules for orientation of the ring system may be used to rule out all grid layouts bar one.

The rules are as follows:

- Maximum number of rings in a horizontal row

This is implemented by iterating over a ring connection table and counting the number of rings where the direction between the rings is identical. The directions which yield the largest number of rings in a line are returned. When multiple ring connection tables are considered, only the combinations of ring connection table and directions that meet this criterion are considered further.

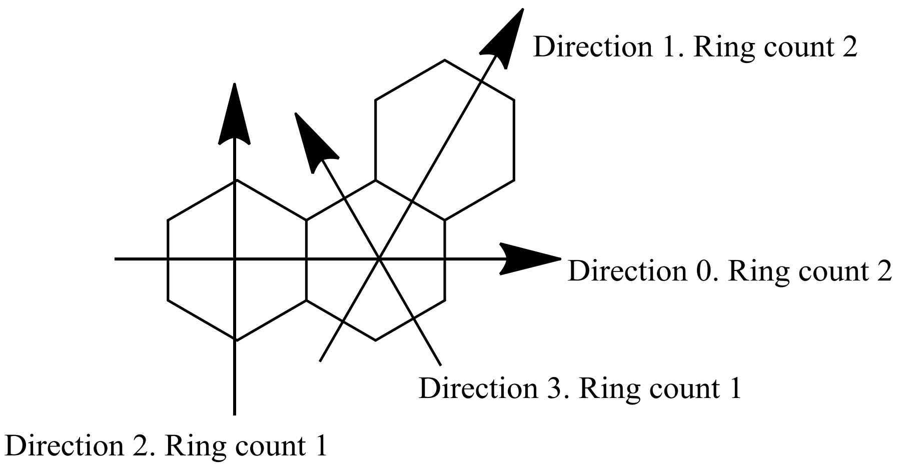

Direction 1. Ring count 2
Direction 0. Ring count 2
Direction 3. Ring count 1
Direction 2. Ring count 1

Figure 3-86 Example of ring counts for different directions

For each applicable ring connection table/direction combination, a grid layout is generated with the given direction defining the horizontal row, i.e. a rotation may be required as compared to the starting ring connection table. If the grid layout has overlapping atoms that grid layout is rejected. This is a minor limitation in OPSIN's implementation, as this is only valid to do when a layout without overlapping atoms actually exists.

- Maximum number of rings in upper right quadrant
- Minimum number of rings in lower left quadrant
- Maximum number of rings above the horizontal row

To check a grid layout against these criteria the grid must be divided up into quadrants. The horizontal divider is defined by the horizontal row and the vertical divider by the mid-point of the horizontal row. The horizontal row is the row with maximum rings in a line. A given grid layout may have multiple rows of rings that meet this requirement hence the division of the system into quadrants must be performed using each possible horizontal row (Figure 3-87).

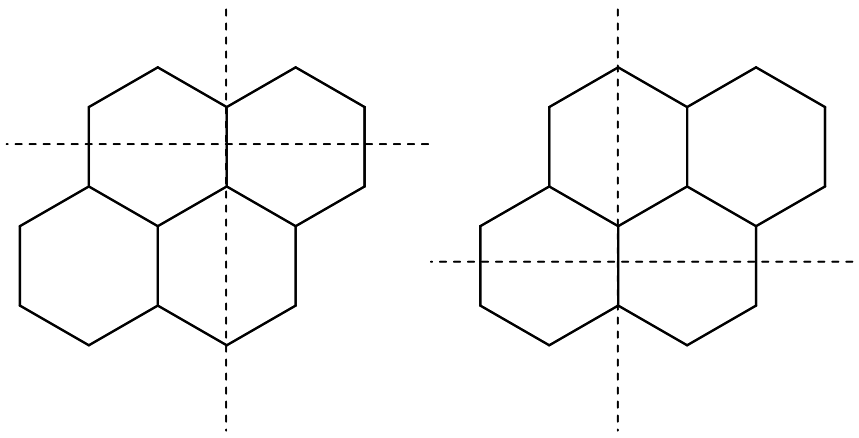


Figure 3-87 Lines showing the quadrants for the two possible horizontal row candidates of this grid layout. The right interpretation is preferred as more rings are in the upper right quadrant.

Counting the occupancy of quadrants is relatively simple with rings contributing a ¼ if the origin is located within a ring, ½ if an axis passes through a ring or otherwise 1 to a particular quadrant. With the quadrant occupancies calculated one can calculate which combinations of grid layout, horizontal row and quadrant give the preferred upper right quadrant.

For each of these combinations, the grid layout is then flipped appropriately such as to place the preferred quadrant in the upper right. The peripheral atoms are then evaluated starting from the uppermost, rightmost ring. These criteria apply entirely to the idealised grid layout and are not affected by how the system would look if drawn. If the candidate ring has no non-fusion atoms the next ring in a clockwise direction is used. The peripheral atoms of the system are then visited starting from most counter-clockwise atom in this ring and proceeding in a clockwise manner around the periphery of the ring system.

Especially for simple fused ring systems, multiple possible numberings (Figure 3-88) are possible and hence the criteria to determine the preferred peripheral numbering must be applied.

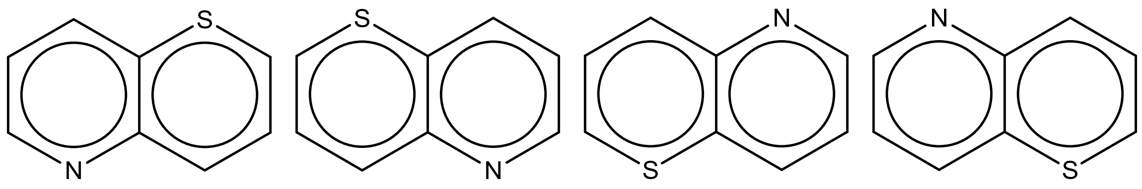


Figure 3-88 Possible orientations of thiopyrano[3,2-b]pyridine without considering peripheral numbering rules. Numbering in all cases would start at the top of the right ring.

3.2.9.12f Peripheral numbering

Preferred peripheral numbering is determined by comparing the lists of possible periphery atoms. Rules include prioritising the list with the earliest heteroatom, the highest priority heteroatom, the earliest fusion carbon atom etc.

OPSIN then iterates over the preferred list numbering the atoms. Numbering increases monotonically except when carbon bridge heads are encountered which are instead labelled with the current number followed by an ascending letter (Figure 3-89).

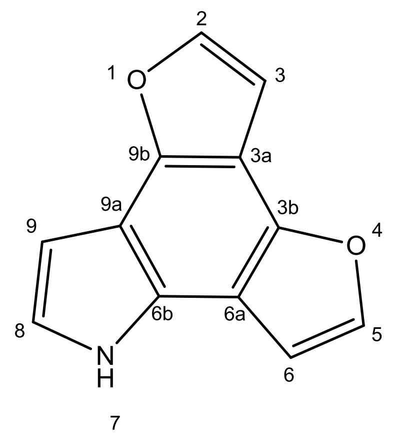
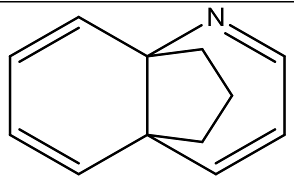


Figure 3-89 7H-difuro[2,3-e:2',3'-g]indole

OPSIN does not implement all of the rules for numbering interior atoms (i.e. those not on the periphery) of fused ring systems meaning incorrect numbering may be produced for these atoms in some cases.

3.2.9.13 Bridges for fused ring systems

Example	 4a,8a-propanoquinoline
General Syntax	locant? bridge bridgeFormingO fusedring

Bridges may be used, in IUPAC nomenclature, on trivially named fused ring systems or those systematically named fused ring systems that could not be named if the bridge were considered part of the fused ring system. The bridge is a non-detachable feature and should be placed adjacent to the fused ring system. A bridge may be a divalent alkyl group, a heteroatom equivalent, a divalent trivial ring or even a mixture of these.

OPSIN only supports the case where a bridge is an alkane or a divalent oxygen atom (or chalcogen equivalent) (Figure 3-90). Practically, this did not appear to be a significant source of failure during evaluation. However, adding further support for bridging nomenclature should not be technically difficult if required in the future.

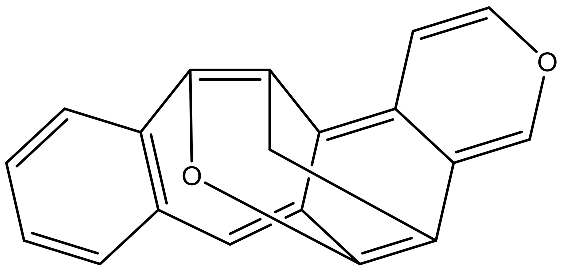
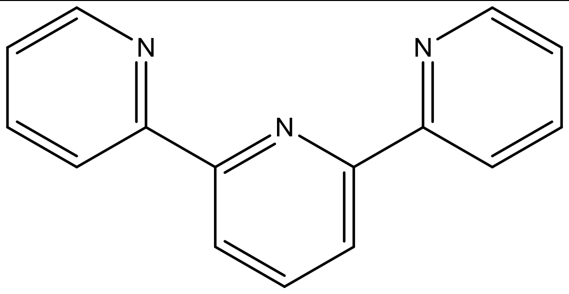


Figure 3-90 6,12-epoxy-5,13-methanobenzo[4,5]cyclohepta[1,2-*f*]isochromene

3.2.9.14 Ring assemblies

Example	 2,2':6',2''-terpyridyl
General Syntax	locant? multiplier ring radicalSuffix?

A ring assembly [76 (Rules A-51 – A-56)] is defined as a system comprising of two or more cyclic systems joined together by single or double bonds such that the number of bonds between the rings is one less than the number of rings. A cyclic system could be any ring or a fused ring system.

For the case when the rings involved are constitutionally identical, the IUPAC recommend specific nomenclature that clearly indicates this relationship between the rings. Two slightly different methods have been recommended for naming such systems: one employs additive operations (no atoms added or removed when forming a bond) and the other employs conjunctive operations (one hydrogen removed from each group when forming a bond) (Figure 3-91).

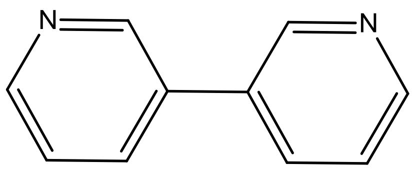


Figure 3-91 3,3'-bipyridine (conjunctive) or 3,3'-bipyridyl (additive)

The naming methods involve using the name of the ring system (conjunctive) or the name of the radical of the ring system (additive) preceded by a Latin multiplier and, where necessary, a locant to indicate which atoms in the rings are connected. As an exception, benzene rings always use the radical name ‘phenyl’, e.g. ‘biphenyl’. An ortho/meta/para locant may be used instead of a normal locant for six-membered rings with bonds that are all in the same relative positions (Figure 3-92).

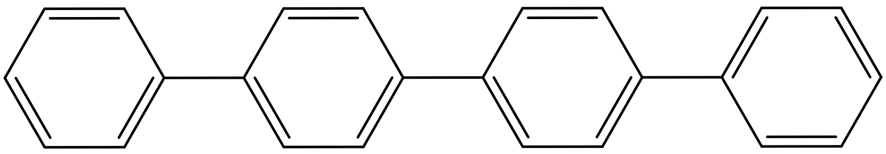


Figure 3-92 p-quaterphenyl

OPSIN has support for all common ring assembly nomenclature. It does not support the use of delta convention to specify a double bond between rings or the new locant system employing superscripts introduced in the provisional recommendations[78] (Figure 3-93). Due to the multitude of ways that are used to represent superscripted characters rather than actual superscripted numbers, it is hoped that this recommendation will not be included in the final recommendations.

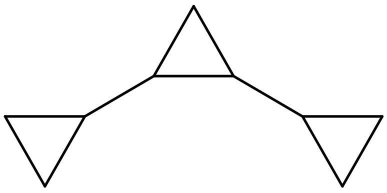
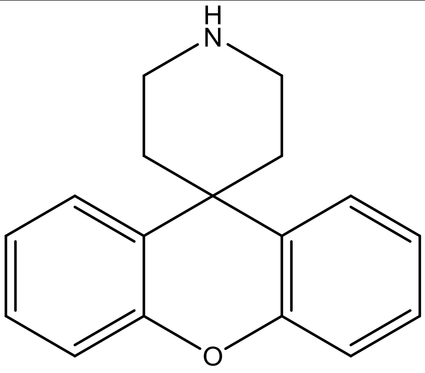


Figure 3-93 1^1,2^1:2^2,3^1-tercyclopropane (new locant system; not OPSIN interpretable) 1,1′:2′,1″-tercyclopropane (current locant system; OPSIN interpretable)

Ring assemblies are handled by first converting an ortho/meta/para locant (if present) into the explicit locant form normally used. Non-detachable features are then resolved onto the ring system before the ring system is duplicated the appropriate number of times. Care is taken to distinguish between features that apply to the individual ring or to the ring assemblage as the latter should not be processed at this stage. The cloned ring systems are then bonded via the atoms indicated by the supplied locants, or by heuristically chosen atoms if no locants are provided.

3.2.9.15 Polycyclic spiro nomenclature

Example	 spiro[piperidine-4,9'-xanthene]
General Syntax	multiplier? spiro openBracket ring (locant ring)+ closeBracket

To name spiro systems made from one or more polycyclic rings, nomenclature employing the names of the constituent ring systems is used[114(Rules Sp-2 – Sp-6)]. The general nomenclature for these systems is to state the number of spiro centres followed by a bracketed section listing the constituent ring systems and the locants of the atoms on them that are involved in spiro fusions (Figure 3-94).

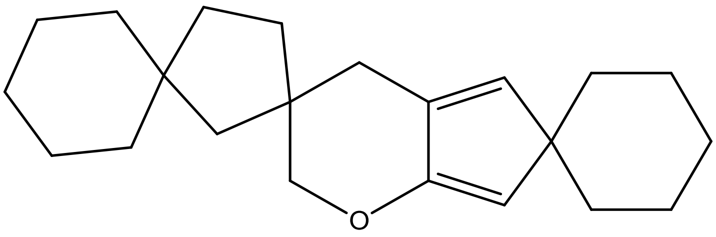
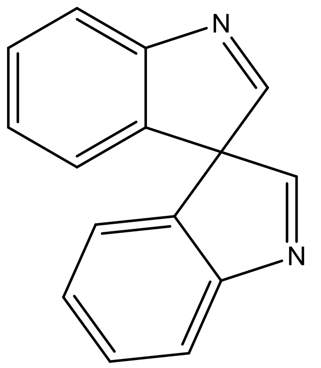


Figure 3-94 2"*H*,4"*H*-trispiro[cyclohexane-1,1'-cyclopentane-3',3"-cyclopenta[*b*]pyran-6",1'''-cyclohexane]

When the ring systems involved are identical a contracted form is employed to avoid repetition (Figure 3-95).



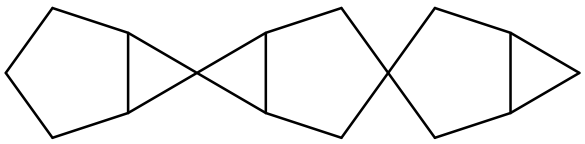


Figure 3-95 Examples of spiro systems with repeated ring systems: 3,3'-spirobi[indole] (left) and 3,3':6',6''-dispiroter[bicyclo[3.1.0]hexane] (right)

OPSIN supports the majority of common polycyclic spiro nomenclature but lacks complete support. OPSIN currently lacks support for systems formed of a mixture of identical and non-identical rings in which the identical rings are mentioned using multipliers e.g. trispiro[1,3,5-trithiane-2,2':4,2'':6,2'''-tris(bicyclo[2.2.1]heptane)]. Another limitation is that locants on ring systems beyond the first should be in square brackets; as OPSIN uses the same expression for rings inside and outside spiro systems this behaviour is supported only in cases where OPSIN allows locants to be enclosed in square brackets outside of a spiro system e.g. 3*H*-spiro[1-benzofuran-2,1'-cyclohex[2]ene] is unsupported.

OPSIN fully supports an older method of naming spiro systems[76(Rule A-42)] which instead has the term 'spiro' and locants indicating the atoms involved in the spiro fusion between the ring systems involved (Figure 3-96).

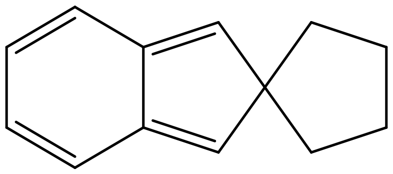


Figure 3-96 2H-indene-2-spiro-1'-cyclopentane

3.2.9.16 D/L stereochemistry

D/L stereochemistry is used to describe how the stereochemistry of a compound compares to the stereochemistry of the two enantiomers of glyceraldehyde; D-glyceraldehyde and L-glyceraldehyde (Figure 3-97).

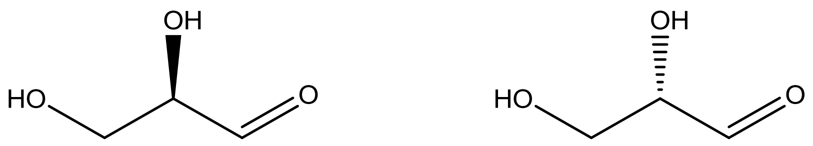
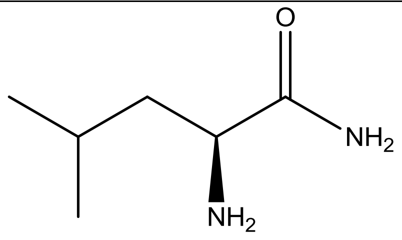


Figure 3-97 D-glyceraldehyde (left) and L-glyceraldehyde (right)

As for both monosaccharides and amino acids one chiral form is significantly more prevalent in nature it may be assumed that when unspecified that this is the form that is referred to (D for monosaccharides and L for amino acids). OPSIN supports this convention by storing such compounds with their stereochemistry defined as in their natural form. D/L stereochemistry can then be simply treated as a modification of this stereochemistry e.g. D- indicates that the stereochemistry of an amino acid should be inverted whilst L- indicates it may be left as is.

Due to this implementation, D/L stereochemistry's rare use in general organic nomenclature e.g. D-α-Amino-β-phenylpropionic acid is unsupported.

3.2.9.17 Amino acid nomenclature

Example	 L-leucinamide
General Syntax	D/L? trivialAminoAcidName suffix?

Amino acid nomenclature[118] provides succinct names for amino acids, amino acid derivatives and polymeric amino acids in peptides. The nomenclature essentially consists of the trivial names for the common amino acids in conjunction with suffix rules that differ slightly from those of general organic nomenclature.

As compared to other carboxylic acids, amino acid nomenclature is only codified for a subset of the suffixes supported in general organic nomenclature. A few quirks that needed to be taken into account when implementing suffixes rules for amino acids were:

- 'ol' and 'al' are valid suffixes (e.g. glycinol) . It should also be noted that on di-acids that these suffixes must be locanted.
- The absence of a suffix is the equivalent of the 'ic acid' suffix

- ‘yl’ means acyl i.e. what ‘oyl’ often means
- Locanted ‘yl’ means add a radical
- ‘o’ may be used to add a radical to the amino nitrogen e.g. glycino

When constructing a peptide the names of the acyl groups of amino acids may be concatenated (Figure 3-98). As brackets are not required, to assure the correct interpretation OPSIN adds implicit brackets (Figure 3-99).

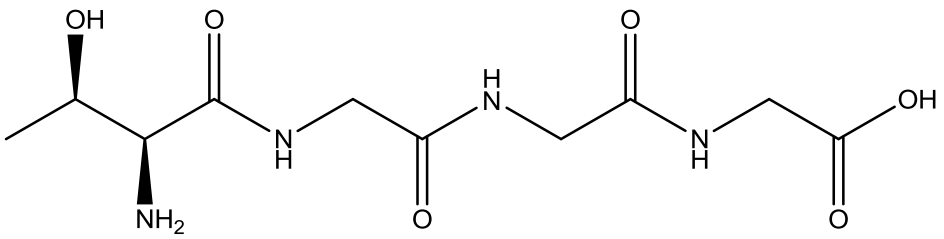


Figure 3-98 threonylglycylglycylglycine

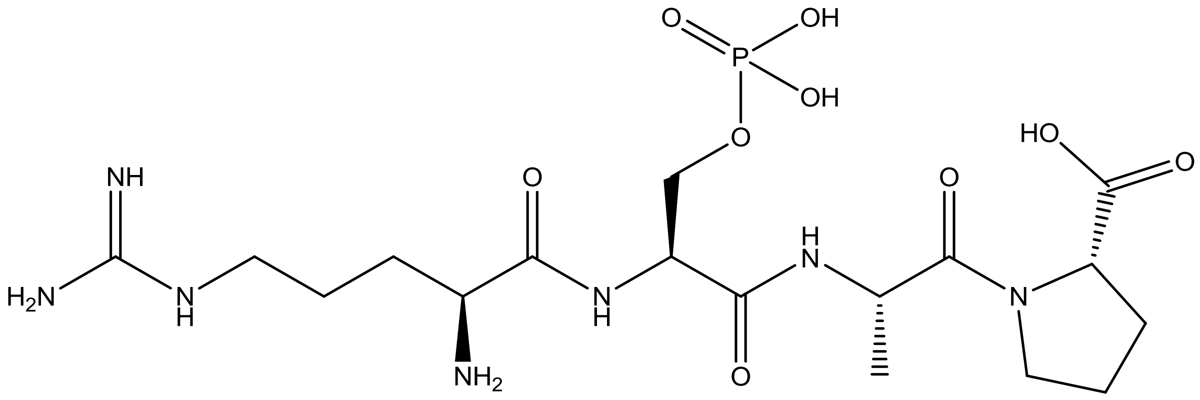
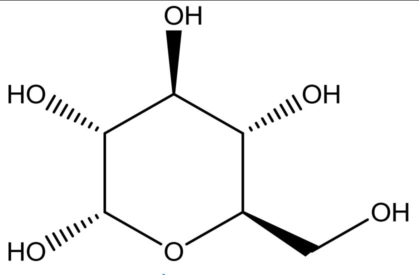


Figure 3-99 ʟ-arginyl-O-phosphono-ʟ-seryl-ʟ-alanyl-ʟ-proline, interpreted as ((ʟ-arginyl-O-phosphono-ʟ-seryl)-ʟ-alanyl)-ʟ-proline

OPSIN supports the majority of common amino acid nomenclature. OPSIN does not support the use of ᴅ/ʟ on achiral amino acids that are made chiral by substitution.

3.2.9.18 Carbohydrate nomenclature

Example	 α-D-glucopyranose
General Syntax	α/β? D/L? carbohydrateStem suffix+

Carbohydrate nomenclature[119] may be employed to more succinctly name saccharides. All aldoses and 2-ketoses (Figure 3-100) of length up to 6 carbons have trivial names with each diastereomer having a different name. A specific enantiomer is indicated by the use of D/L in front of the trivial name, which relates the configuration of the highest-numbered carbon stereocentre (the configurational atom) to that of D/L- glyceraldehyde (Figure 3-101, *cf.* Section 3.2.9.16).

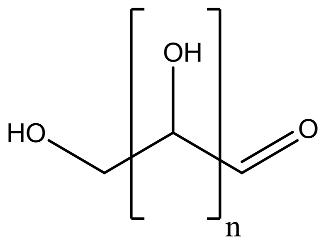
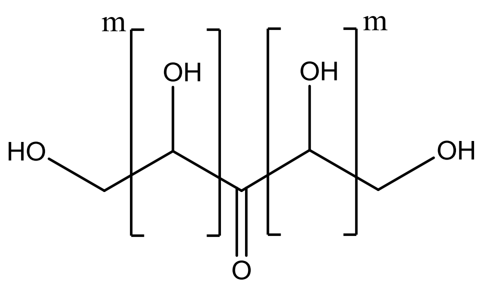



Figure 3-100 Structure of aldoses (left) and ketoses (right) where n is 1 or more and m is 0 or more. A 2-ketose is one in which at least one of the m's is 0.

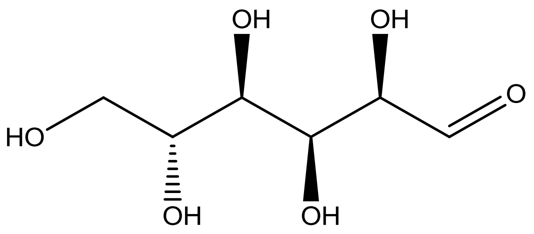
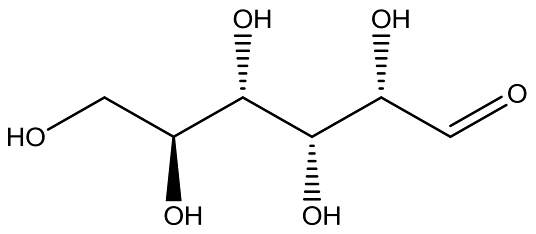



Figure 3-101 D-glucose (left) and L-glucose (right)

To create carbohydrate derivatives either the trivial name of the carbohydrate without the terminal 'se' or a systematically defined stem may be used (Section 3.2.9.18a). This is then followed by suffixes indicating additions or modifications to the chain.

Monosaccharides most commonly are found in a cyclic form as a hemiacetal or a hemiketal so one of the most common suffixes employed indicates the ring size formed when the carbohydrate cyclises. For example furanose for a 5-membered ring or pyranose for a 6-membered ring (Figure

3-102). Cyclisation forms an additional stereocentre referred to as the anomeric centre. The configuration of this centre is specified using either α or β. These specify the relationship between the stereochemistry at the anomeric reference atom and the anomeric centre. The anomeric reference atom and configurational atom are always synonymous unless the carbohydrate stem has been systematically defined.

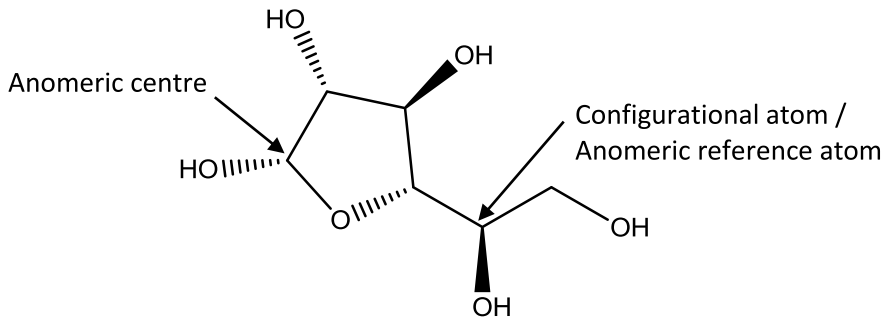
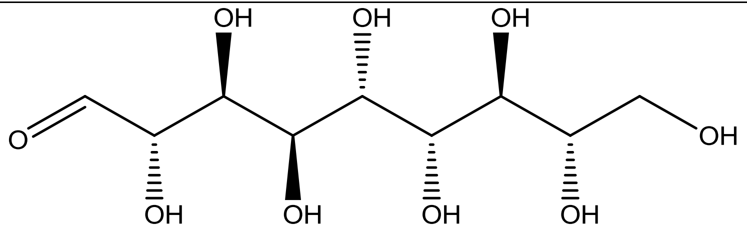


Figure 3-102 α-D-galactofuranose. D-galactose cyclised to form a 5 member ring.

OPSIN supports cyclising all IUPAC endorsed trivial carbohydrate names but does not currently support cyclisation of systematically defined stems, or any other suffixes.

3.2.9.18a Systematic carbohydrate chains

Example	 L-*ribo*-D-*manno*-nonose
General Syntax	(D/L? configurationalPrefix)+ chainLength suffix+

Monosaccharides lacking trivial names may be named using configuration prefixes derived from the names of the trivial aldoses. These prefixes specify that the defined stereocentres have the same stereochemistry as the aldose from which the prefix was derived. The number of stereocentres the prefixes define should be exactly the same as the number of stereocentres that are in the sugar. Of note from an implementation perspective the configuration prefixes refer to the final structure of the sugar e.g. after subtractive nomenclature has been performed (Figure 3-103).

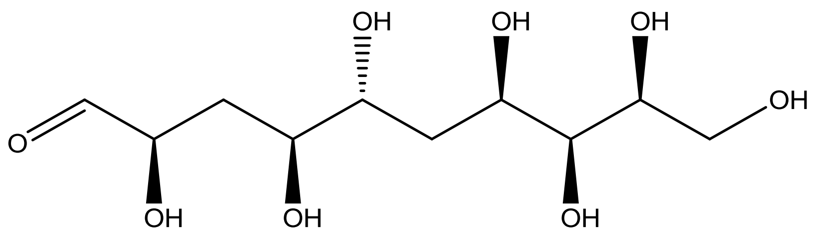
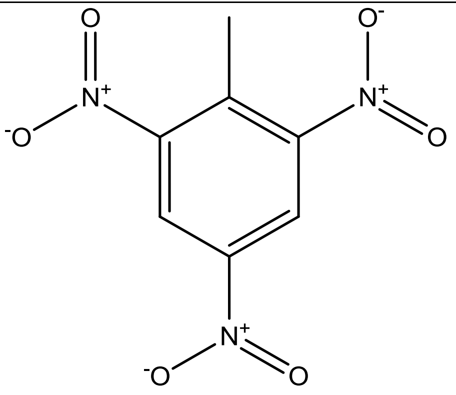


Figure 3-103 3,6-Dideoxy-L-*threo*-L-*talo*decose. *threo* specifies the configuration at 2 centres and *talo* at 4. A decose has 8 stereocentres but two are removed by removal of hydroxyl groups.

Any trivial carbohydrate chain name may be specified using this nomenclature e.g. D-glucose = D-*gluco*-hexose.

3.2.10 Structure assembly

3.2.10.1 Substitutive nomenclature

Example	 2,4,6-trinitrotoluene
General Syntax	(locant? multiplier? substituent)+ parentGroup

The substitutive operation is the most common method of connecting fragments in organic chemical nomenclature. A substitutive operation involves the replacement of one, or more, hydrogen atoms by another fragment. The number of hydrogens replaced is determined by the valency of the radical on the replacement fragment e.g. ‘yl’ =1, ‘ylidene’ =2 etc. Substitutive nomenclature is performed recursively on the substituents/brackets respecting bracketing so as to ensure the correct groups are substituted.

OPSIN supports two special cases of substitutive nomenclature. Perhalogeno terms e.g. ‘perchloro’ indicate that all substitutable hydrogens have been replaced by the indicated halogen (Figure 3-104).

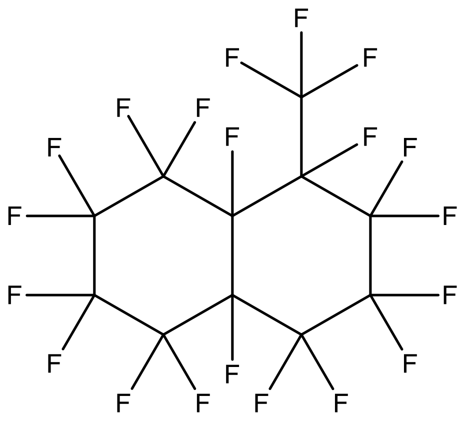


Figure 3-104 perfluoro(decahydro-1-methylnaphthalene)

The other special case is for bridging substituents such as epoxy, epithio and methylenedioxy. Unlike fused ring bridges these may be applied to systems that are not fused rings and additionally are often treated as detachable prefixes. Bridging substituents differ from normal substituents in that they may be preceded by two locants indicating the two atoms to which the bridging substituent attaches (Figure 3-105).

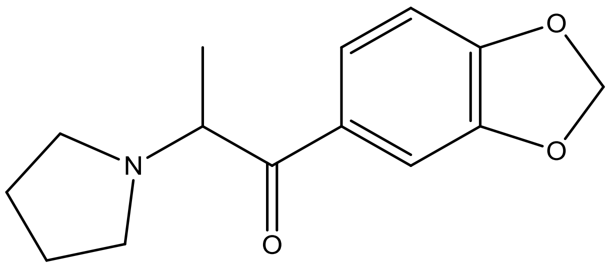


Figure 3-105 3',4'-Methylenedioxy-α-pyrrolidinopropiophenone. The substitution of the benzene ring by methylenedioxy yields the 1,3-Benzodioxole ring system.

OPSIN supports alphanumeric (e.g. '1', '3a'), Greek (e.g. 'beta'), element symbol locants (e.g. 'N') and element symbol locants in combination with alphanumeric locants (e.g. 'N^4' or '4-N'). All locant types may contain primes. Element symbol locants are assigned algorithmically using a series of empirically defined heuristics that reproduce the labelling IUPAC has specified for certain nitrogen containing suffixes. Locants that combine an alphanumeric component with an element symbol locant are not assigned, as in most cases such locants will never be referred to. Instead, when such a locant is requested, the atom corresponding to the alphanumeric part of the locant is looked up and then a search for an atom that is connected either directly or through atoms without alphanumeric locants is initiated to find the atom matching the element symbol portion of the locant. The expected element symbol locant may differ from that assigned when the molecule was considered as a whole as the element symbol locant will be based off just the atoms connected to the point in the molecule being investigated (*cf*. the two different atoms referred to as N′ in Figure 3-106).

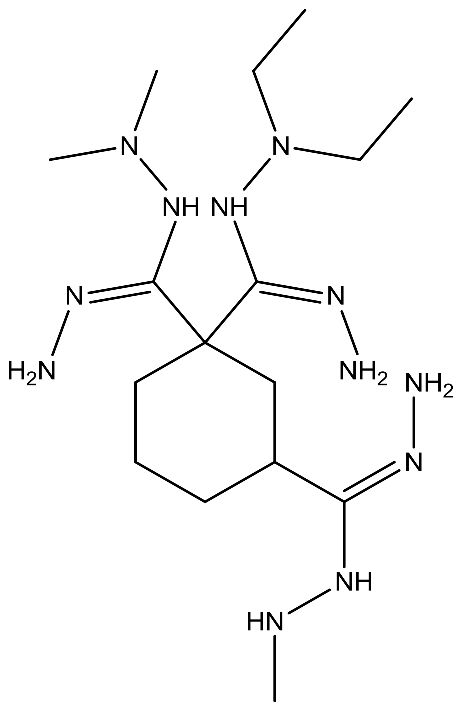


Figure 3-106 1-N′,1-N′-diethyl-1-N′′′ ′′,1-N′′′ ′′,3-N'-trimethylcyclohexane-1,1,3-tricarbohydrazonohydrazide

3.2.10.2 Additive nomenclature

An additive operation involves the joining of two fragments together without loss of atoms. In the context of joining fragments this typically applies to the bonding of radicals together (e.g. Figure 3-107). Some operations in functional class nomenclature (Section 3.2.10.4) are also formally additive operations.

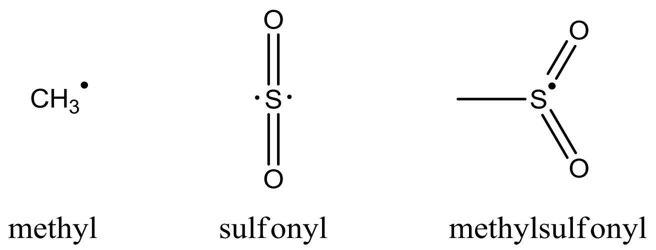


Figure 3-107 Methyl and sulfonyl are combined via an additive operation to create the prefix methylsulfonyl

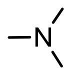
As additive operations may only occur between substituents that are adjacent within a chemical name, OPSIN performs additive operations prior to performing substitutive operations. Without this ordering of operations some names that are not perfectly formed, but are in common parlance considered unambiguous, e.g. methylsulfonylcyclohexane become ambiguous. In this case the methyl may be interpreted as a substituent of the cyclohexane if it is not first additively bonded to the sulfonyl.

Implementation of this nomenclature is significantly complicated by ambiguity in some substituents as to whether or not they are multi-valent radicals (Figure 3-108). The left-hand interpretation for these two substituents implies a substitutive operation interpretation in which a double bond is formed.

$H_2C=$ $-CH_2-$ $HN=$ $-NH-$

Figure 3-108 Interpretations of methylene (left) and imino (right)

Another ambiguity that affects a small number of substituents relates to the valency of the radicals that the substituent possesses (Figure 3-109). This ambiguity occurs most often in multiplicative nomenclature. The draft 2004 recommendations now only recommend the name nitrilo for the left interpretation in Figure 3-109.

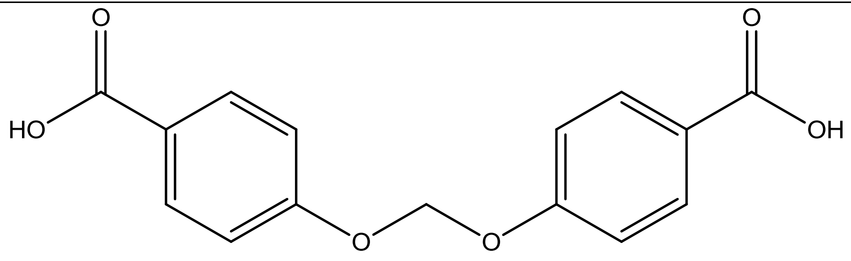


$=N-$ $-N=$

Figure 3-109 Interpretations of nitrilo in general organic nomenclature. The left interpretation is preferred.

Disambiguation can be achieved in most cases by examining the adjacent substituent e.g. is it a multi-valent radical.

3.2.10.3 Multiplicative nomenclature

Example	 4,4'-methylenedioxydibenzoic acid
General Syntax	locant? (substituent multiplier)+ parentGroup

Multiplicative nomenclature[76(Rules C-72 to C-74)] is used when a structure may be assembled from multiple identical components. All substituents involved are multi-valent radicals with additive operations connecting the substituents.

Multiplicative nomenclature is implemented as a special case of additive nomenclature. It is detected by the presence of a multi-valent radical group followed by a multiplier equal to the valency of the multi-valent radical group. Once multiplicative nomenclature has been detected, groups are joined from left to right until the parent group is reached.

Additionally a special case is required to allow the case where a substituent that is not obviously a multi-valent radical acts as one e.g. the benzylidene in Figure 3-110.

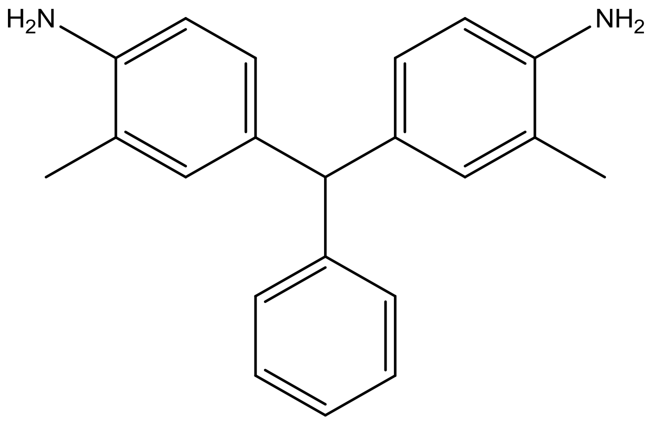
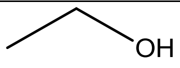


Figure 3-110 4,4'-benzylidenedi-o-toluidine

3.2.10.4 Functional class nomenclature

Example	 ethyl alcohol
General Syntax	groupOrSubstituent functionalTerm

Functional class nomenclature involves a group, often a substituent, followed by a class name. Traditionally, in the case where the group is a substituent, this nomenclature was called radicofunctional nomenclature. OPSIN has specific rules for dealing with different types of functional class nomenclature which roughly parallel the word rules mentioned in Section 3.2.6. For example the same code may be used for all “carbonyl derivatives” e.g. oximes, hydrazones, semicarbazones and imides.

Some class names may be related to a chemical structure that will either be bonded onto the preceding fragment (e.g. ‘cyanate’ or ‘ketone’) or replace an atom on a preceding fragment (e.g. ‘oxime’). Some of these fragments may even be substituted and hence are best treated as normal groups prior to being incorporated into the preceding fragment (Figure 3-111).

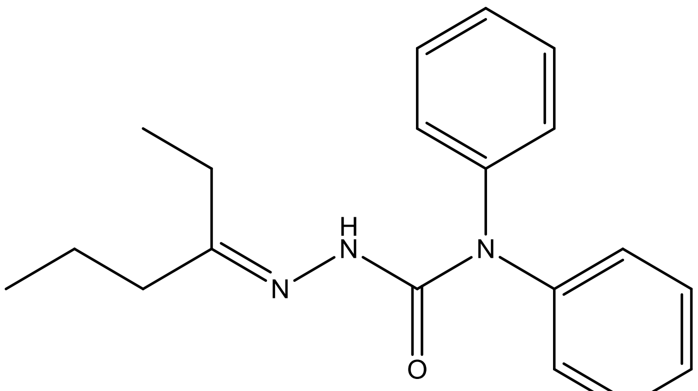


Figure 3-111 hexan-3-one 4,4-diphenylsemicarbazone

Other class names such as ‘ester’ (Figure 3-112) or ‘acetal’ (Figure 3-113) are purely used to determine what operation needs to be performed on the groups that are present.

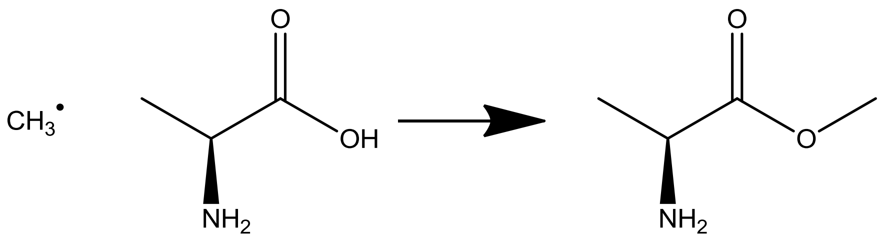


Figure 3-112 L-alanine methyl ester, constituent parts (left) and final structure (right)

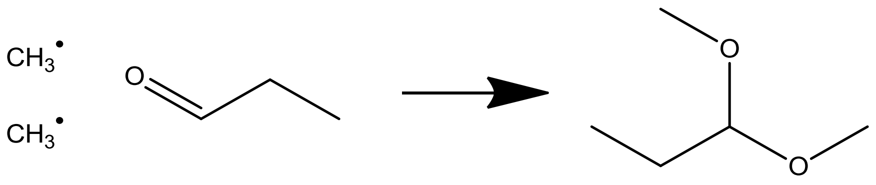
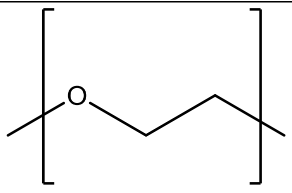


Figure 3-113 propanal dimethyl acetal, constituent parts (left) and final structure (right)

3.2.10.5 Structure-based polymer nomenclature

Example	 poly[oxyethylene]
General Syntax	poly substituent+

Polymers may be represented in IUPAC nomenclature by naming the repeat unit preceded by ‘poly’[120]. With the addition of only a few special cases, OPSIN is able to support the nomenclature used to describe a repeat unit as part of its general handling of additive nomenclature.

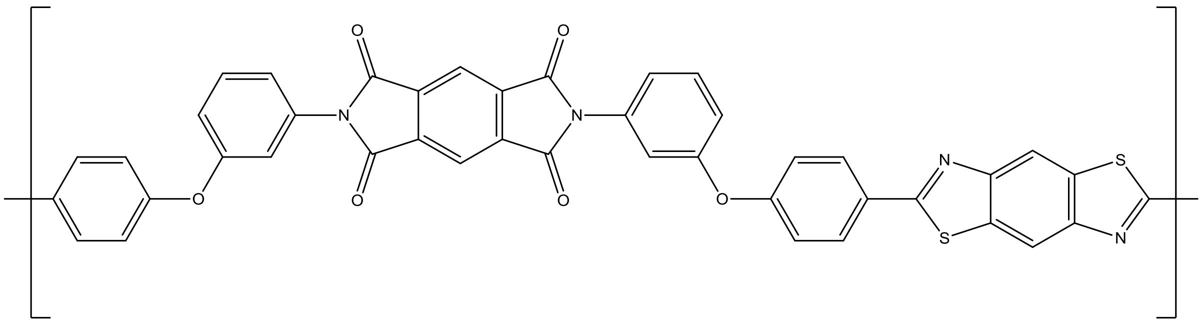


Figure 3-114 poly[(benzo[1,2-d:4,5-d']bis[1,3]thiazole-2,6-diyl)-1,4-phenyleneoxy-1,3-phenylene(1,3,5,7-tetraoxo-1,2,3,5,6,7-hexahydrobenzo[1,2-c:4,5-c']dipyrrole-2,6-diyl)-1,3-phenyleneoxy-1,4-phenylene]

A special case was required to handle the fact that 'imino' and 'methylene' are used nearly exclusively as linkers in polymer nomenclature whilst in general nomenclature they can often refer to a double bonded atom (Figure 3-115).

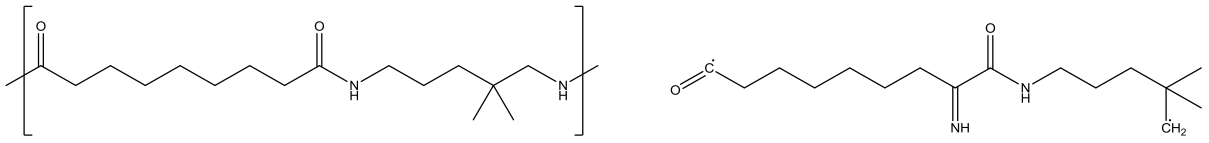


Figure 3-115 OPSIN's interpretations of poly(imino-2,2-dimethylpentamethyleneiminoazelaoyl) (left) and imino-2,2-dimethylpentamethyleneiminoazelaoyl (right)

Another special case was required for those groups with three or more connections that only have two in polymer nomenclature (Figure 3-116)

=N−   −N=

Figure 3-116 Interpretations of nitrilo in polymer nomenclature. Note that for nitrilo this is in direct contradiction with the 2004 draft recommendations which specify that nitrilo should refer only to the interpretation with three connections *cf*. Figure 3-109

3.2.11 Kekulisation

During the assembly of fragments, double bonds on atoms in rings are not explicit. Instead they are represented using the flag indicating that the atom may be involved in a π system, that was mentioned previously in connection with SMILES reading (Section 3.2.8) and operations that add/remove hydrogen (Section 3.2.9.7b). Before performing kekulisation this flag is removed from any atoms which by forming a double bond would end up in an unusual valency. It is also removed from atoms that are adjacent only to atoms that may not form double bonds.

For kekulisation to be successful there must be an even number of atoms possessing the flag. If there are an odd number of atoms, an atom with the flag is selected via a series of heuristics to be eliminated from use in double bond formation. These heuristics are in order of priority:

- An atom that was indicated as having hydrogen in the original fragment
- An atom that is adjacent to only one atom with the flag set
- An atom adjacent to two bridgehead atoms
- A heteroatom
- An arbitrarily chosen atom

The algorithm adds double bonds first to atoms that have only one neighbour to which they are capable of being double bonded. Subsequently bonds in which at most one atom is a bridgehead may be considered followed finally by bonds in which both are bridgeheads. A more rigorous solution allowing backtracking when placing double bonds, such that an earlier misplaced double bond will not prevent kekulisation, is a possible future improvement. Nonetheless, except in cases where the position of indicated hydrogen has been underspecified and the name is hence ambiguous, cases of this algorithm failing are extremely rare.

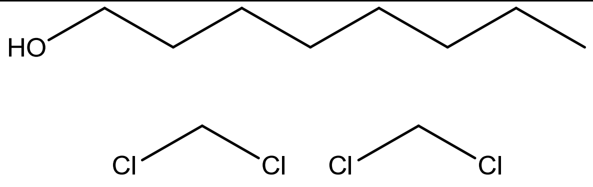
3.2.12 Valency checking

Once all the fragments have been assembled a check is performed on the valency of each atom. The valency is checked either against the highest known stable valency for that atom's element/charge, or against the Lambda Convention specified valency (taking into account protons added/removed by charge modifying suffixes). If a valency check fails, then the name is rejected.

A rationalisation for the decision to reject such structures rather than producing a hypervalent interpretation is that in substitutive nomenclature it is impossible to generate a hypervalent structure (without the Lambda Convention) as only as many hydrogens as are present on the atom when in its standard valency may be substituted. This means that a name that produces a hypervalent structure is not only chemically suspect but also formally incorrect.

3.2.13 Application of stoichiometry

3.2.13.1 Mixtures

Example	 methylene chloride compound with octanol (2:1)
General Syntax	component (compound with)? component+ stoichiometry

Mixtures may be specified by stating the components of the mixture followed by indication of stoichiometry. Often the components are separated by a term like 'compound with'. OPSIN has a small list of terms that are accepted between chemical names and subsequently ignored to achieve this.

Indication of mixture stoichiometry is recognised and stripped from the name prior to tokenisation/parsing. Once word rules have been assigned, the indicated stoichiometry is added as an attribute of each top level `wordRule`. As top level word rules correspond to separate structures, there is expected to be stoichiometry indication for as many components as there are top level word rules. Once processing of the word rules has been completed their contents are multiplied out appropriately.

3.2.13.2 Charge balancing

Example	Mg^{2+} Cl^- Cl^- magnesium chloride (fully specified this name would be magnesium(2+) dichloride)

Compounds described in the chemical literature are typically intended to be overall charge neutral. As a result indication of explicit stoichiometry is often omitted. The problem is further complicated by metals, which often have their charge omitted. If the compound is formed of more than one component and is not charge neutral, OPSIN goes through a series of heuristics to attempt to balance the charge on the compound. These are:

- If a metal is uncharged and has fewer bonds than its typical oxidation state, it indicates that it is a candidate for being made into a cation.

- Potential cationic metals are set to their typical charges (Figure 3-117)

$$Na^{+} \quad Cl^{-}$$

Figure 3-117 sodium chloride. The sodium is set to its standard charge of +1 resulting in a neutral compound

- If setting the metal to its typical charge doesn't satisfy the charge imbalance a higher charge is tried if a higher charge is known to be possible (Figure 3-118)

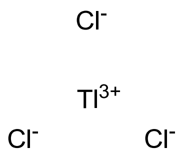


Figure 3-118 thallium trichloride. Thallium is typically thallium(1+) but as there are known to be three chlorides thallium(3+) is assumed.

- Where stoichiometry is undefined and the choice of component/s to multiply is unambiguous, components are multiplied. Components may only be multiplied by integers (Figure 3-119).

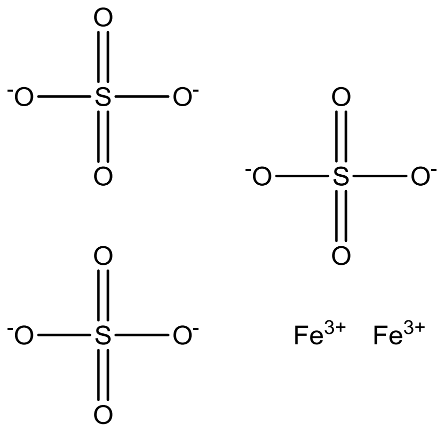


Figure 3-119 iron(3+) sulfate. Typically only one component needs to be multiplied, but in some cases such as this both are.

- A metal has its charge set lower than its typical charge (Figure 3-120)

$$Mg^{+} \quad Cl^{-}$$

Figure 3-120 magnesium monochloride. As there is explicitly only one chloride the number of chlorides may not be adjusted. Hence the charge on the magnesium is adjusted

- A salt is neutralised[76(Rule C816.4)] (Figure 3-121)

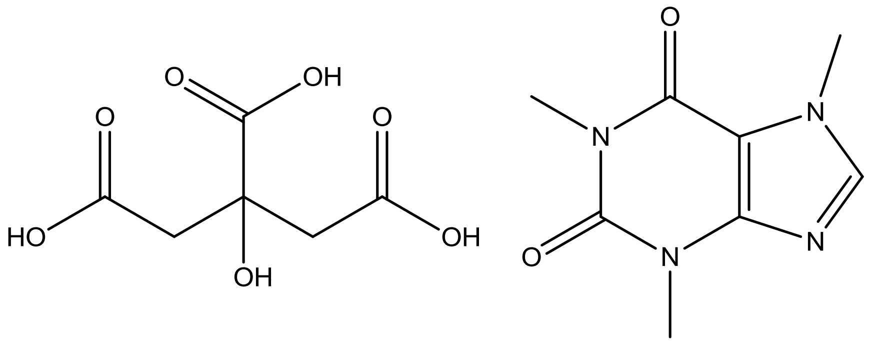


Figure 3-121 caffeine citrate. Citrate in isolation would be treated as a tri anion but as there is another compound present it is treated as if it were citric acid.

3.2.14 Stereochemistry handling

3.2.14.1 Detection of stereocentres

Tetrahedral (e.g. Figure 3-122) and double bond (Figure 3-123) stereochemistry are commonly found in organic chemicals.

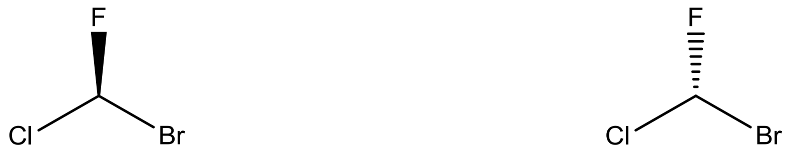


Figure 3-122 (R)-bromochlorofluoromethane (left) and (S)-bromochlorofluoromethane (right)

Figure 3-123 (E)-but-2-ene (left) and (Z)-but-2-ene (right)

As when unambiguous to do so locants are often omitted from stereochemistry prefixes, any rigorous solution to this area must be capable of detecting stereocentres. For this purpose, OPSIN employs a derivative of the InChI canonicalisation algorithm[121,122] to label atom environments. Hydrogen are made explicit prior to stereocentre perception and hence do not present a problem. Higher bond orders are handled, in an analogous way to the Cahn-Ingold-Prelog (CIP) sequence rules[123–125], by treating all bonds as if they were single and adding additional atoms to the atoms at both ends of the higher order bond. The end result is that each constitutionally distinct atom environment is given its own environment number.

These atom environments are then used to identify true stereocentres[126] i.e. stereocentres that do not require the existence of other stereocentres in the molecule to be stereocentres. For detecting tetrahedral stereocentres, a list of atoms to consider is generated by finding those that correspond to known atom/bond configurations that may be tetrahedral stereocentres (Figure 3-124). This approximately corresponds to the stereocentres detected by InChI[122(Table 8)].

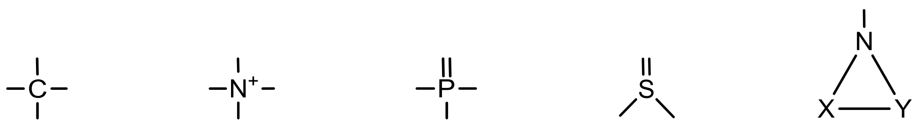


Figure 3-124 Examples of tetrahedral stereocentres recognised by OPSIN. X and Y are two atoms in different environments that are bonded together.

OPSIN ignores those centres that nominally meet these criteria but in reality would not be stereocentres due to simple resonance or tautomerism. Again this approximately corresponds to the specification of InChI although the case depicted in Figure 3-125 is not explicitly mentioned in the specifications.

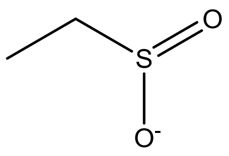


Figure 3-125 Due to resonance this structure is achiral

The list of true stereocentres is then produced by checking that all atoms neighbouring the potential stereocentre are in different atom environments.

Double bond stereocentres are found by analysing the atom environments at either end of a double bond. Each atom in the double bond is expected to be bonded to a total of 3 atoms unless the atom is nitrogen in which case 2 is acceptable with the third “atom” being a lone pair.

OPSIN does not currently detect cumulene stereochemistry although doing so would not be technically challenging.

If an atom in a fragment has defined stereochemistry but is not identified as a stereocentre this information is removed as it is assumed that the atom is no longer a stereocentre in the final molecule e.g. substitution of a hydrogen atom may have made two substituents equivalent.

3.2.14.2 Applying stereochemistry

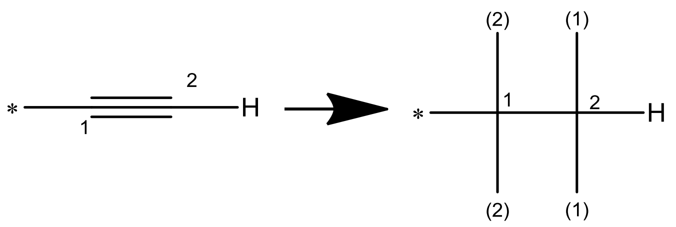
OPSIN performs stereochemistry operations in the order: locanted stereochemistry, carbohydrate stereochemical prefixes, unlocanted stereochemistry; whilst tracking which stereocentres have had their configuration set. As it is not uncommon for a structure with implicit stereochemistry to have this stereochemistry overridden, these cases are not considered as having set the configuration of the stereocentres. OPSIN currently considers five distinct types of stereochemistry R and S, E and Z, *cis* and *trans*, alpha and beta, and carbohydrate stereochemistry.

3.2.14.2a R/S/E/Z stereochemistry

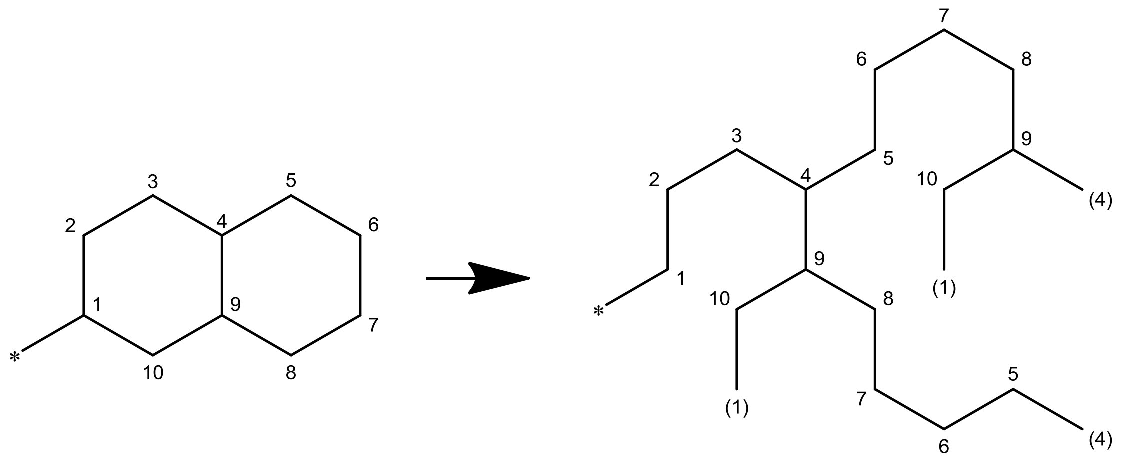
For R, S, E and Z stereochemistry once an appropriate stereocentre has been identified the "ligands" i.e. connected atoms, must be ranked using the CIP system. OPSIN's implementation includes support for rules 1 (higher atomic number preferred to lower) and 2 (higher isotope preferred to lower), which deal with constitutional differences between ligands. A failure is reported if ligands cannot be distinguished.

The 1982 revision to the CIP system[124] introduced the concept of hierarchical digraphs. A hierarchical digraph is an acylic graph representation of the bonding within a ligand. The transformation from the connection table of a ligand to a digraph involves two transformations:

- Bonds of order greater than 1 are represented as single bonds with attached duplicated atoms (called ghost atoms) e.g.:



- Bonds that join to an atom previously visited by that branch of the digraph instead join to a ghost atom which is not further bonded e.g.:



Rules 1 and 2 involve comparing the hierarchical digraphs for each ligand with rule 2 only being invoked if rule 1 fails to distinguish the ligand. This comparison starts from the first layer of atoms from the stereocentre. Evaluation proceeds on a layer by layer basis with a subsequent layer only being investigated if the prior layer failed to distinguish the ligands. It should be noted that the ordering of atoms in each layer is determined by the priority of atoms in the previous layer, and only when a tie is encountered by the relative priority of the atoms within the layer.

OPSIN's implementation is notable in that it only lazily evaluates the digraph. As typically ranking may be determined within the first couple of layers, this approach is computationally faster and more memory efficient, especially for larger molecules (Figure 3-126).

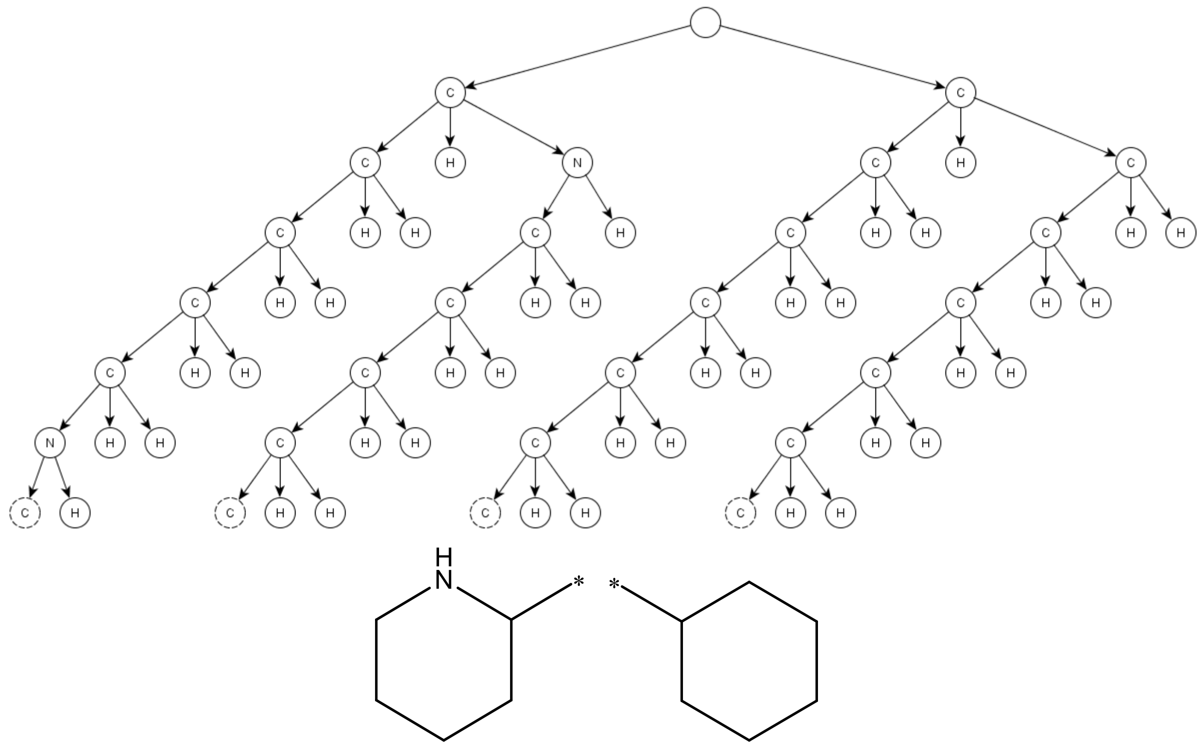


Figure 3-126 Hierarchal diagraph for piperidin-2-yl and cyclohexyl ligands. The two ligands are distinguished by OPSIN at the 2nd level as [N,C,H] has higher priority than [C,C,H] with no further enumeration of the digraph required.

OPSIN also implements a corner case in rule 1 (rule 1b[125]) in which two ligands may be constitutionally different but have identical hierarchical digraphs (Figure 3-127). In this case ghost atoms must be distinguished from non-ghost atoms and the position of the atom the ghost atom is a duplicate of in the digraph is taken into account.

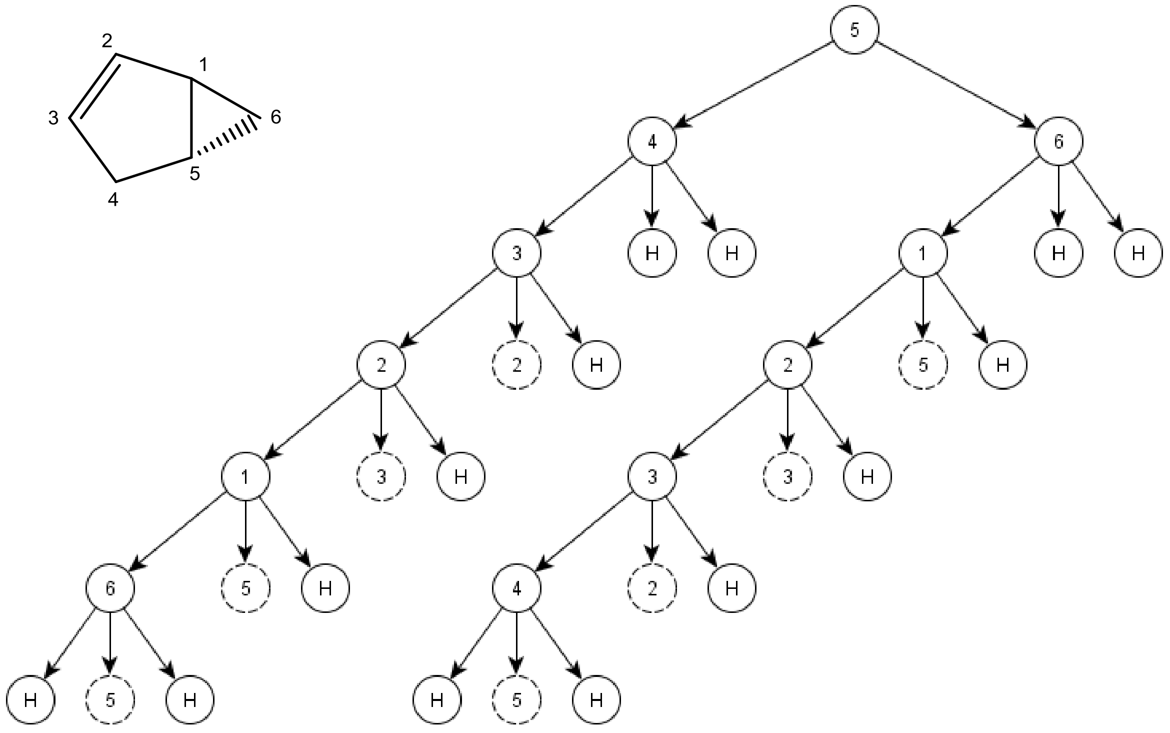


Figure 3-127 (5S)-bicyclo[3.1.0]hex-2-ene

From the combination of ordered ligands and a stereodescriptor, e.g. R/S/E/Z, it is then simple to define the stereochemistry of a tetrahedral centre or double bond.

3.2.14.2b Cis/trans stereochemistry

Cis and *trans* are initially interpreted as referring to the relative stereochemistry of two substituents on a ring. OPSIN does not have general support for detecting pseudo-asymmetric atoms but has support for such stereocentres in this particular case. A ring system is investigated to find tetrahedral atoms that either have one hydrogen or are connected to a fragment outside of the ring system. If there are exactly two of them, their configuration may be set to be relatively *cis* or *trans*. To do this the smallest set of smallest rings is calculated, allowing a list of all bonds not involved in fusions to be compiled. From these bonds two paths joining the stereocentres should be discoverable (Figure 3-128). This knowledge of the positioning of atoms at one stereocentre relative to the positioning of atoms at the other stereocentre allows OPSIN to construct descriptions of the stereochemistry that assure that the two centres will be *cis*/*trans* to each other. If one atom has predefined stereochemistry, care is taken to leave that stereocentre as defined and have the other stererocentre's configuration be relative to the predefined stereocentre.

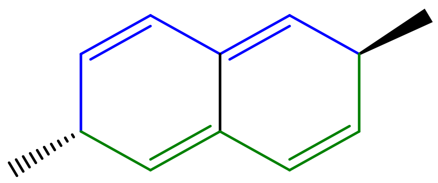


Figure 3-128 *trans*-2,6-dimethyl-2,6-dihydronaphthalene. Coloured atoms show the paths defining the periphery of the molecule. By using the atoms at either end of the blue path and at either end of the green path in the same place in the generated stereochemistry descriptions one can tie the configuration of the two stereocentres together.

OPSIN also allows *cis*/*trans* to be used as an alternative to E/Z to specify double bond stereochemistry but only in the special case where one group at either end of the double bond is hydrogen. Without this criterion, it is formally ambiguous as to which groups are being defined as *cis*/*trans* to each other.

3.2.14.2c Alpha/beta stereochemistry

Alpha/beta stereochemistry is used to indicate on which side of a plane a group is positioned. OPSIN only current supports alpha/beta stereochemistry in conjunction with natural product nomenclature[127 (RF-10)]. In natural product nomenclature, a particular depiction of the molecule is designated as the preferred orientation and it is with respect to this that alpha/beta stereochemistry is defined. OPSIN encodes this information by associating each natural product that supports alpha/beta stereochemistry with a list of the peripheral atoms of the natural product when read in a clockwise direction. The positioning within this list of the adjacent periphery atoms to the stereocentre, the atom to which alpha/beta is referring and the alpha/beta itself, is sufficient to define the stereo configuration (Figure 3-129).

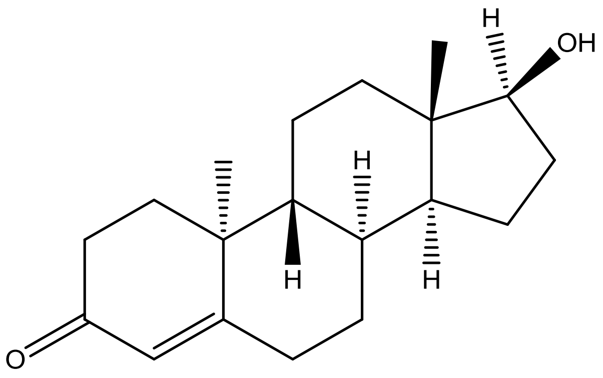


Figure 3-129 17β-Hydroxy-8α,9β,10α-androst-4-en-3-one

3.2.14.2d Carbohydrate stereochemistry

Carbohydrate stereochemistry is only employed on the systematic carbohydrate stems described previously in Section 3.2.9.18a. OPSIN's vocabulary has these carbohydrate stems with their stereocentres configured such that the hydroxyl groups would point right on a Fischer projection (Figure 3-130). The configuration prefixes can then be simply implemented as a list of 'r's and 'l's indicating whether or not the configuration at each centre should be retained or flipped. For example D-*gluco* is expressed as "r/l/r/r". To be valid a carbohydrate name must have every stereocentre in its stem, which still exists after substitutive and subtractive nomenclature operations have been applied, defined by configurational prefixes.

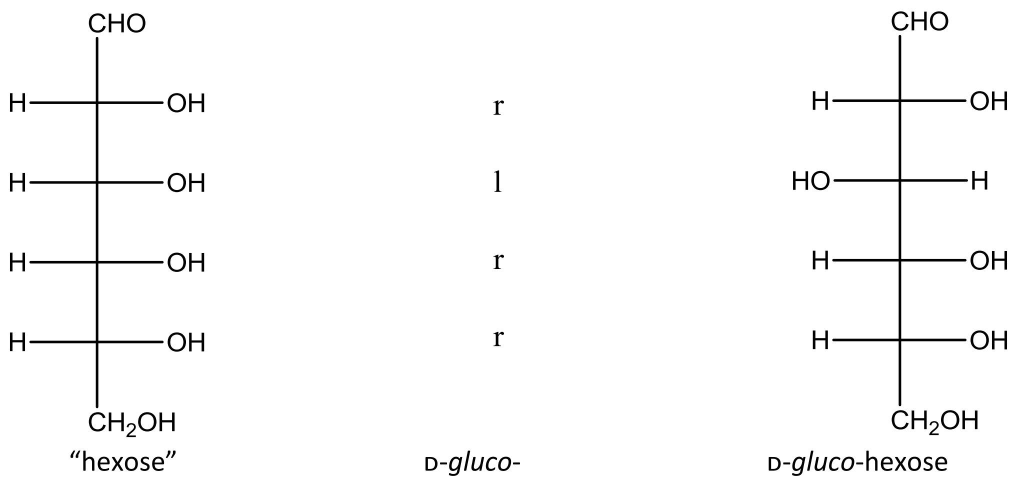


Figure 3-130 Fischer projection for D-*gluco*-hexose showing the method of constructing the stereochemistry for the complete name

3.2.15 Ambiguous and formally incorrect chemical names

When a chemical name is underspecified e.g. lacking sufficient brackets or locants it may become ambiguous and formally describe multiple structures. OPSIN has been empirically tuned to attempt to generate the interpretation of a name that is most likely in common usage, with an implicit assumption that an input chemical name is intended to describe a particular structure. This is very similar to one of Brecher's principles for a chemical nomenclature interpretation system[106]: "The meaning of logically ambiguous names is determined by common usage". The addition of implicit brackets or spaces may be sufficient to give a formally ambiguous or highly unlikely name, an unambiguous and likely interpretation. Heuristics for making these alterations are dealt with in the following subsections.

3.2.15.1 Implicit bracketing

Implicit bracketing is employed by OPSIN in cases where substitution onto the rightmost group, in the current scope, of a chemical name is not intended (Figure 3-131).

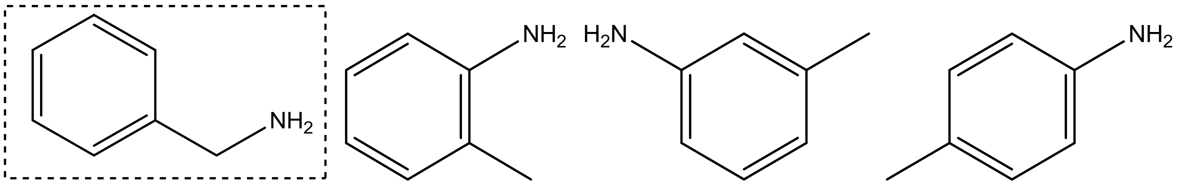


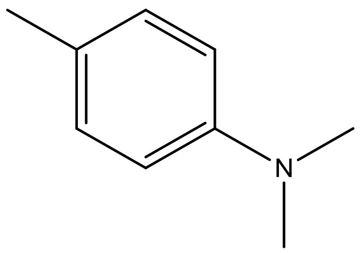
Figure 3-131 Allowed interpretations of aminomethylbenzene. The boxed interpretation is produced by OPSIN by implicitly bracketing the name to (aminomethyl)benzene

Figure 3-131 depicts four structures consistent with the name, aminomethylbenzene. There is only one possible structure where the aminomethyl is a substituent on the benzene ring, whereas if the amino and methyl groups are direct substituents of the ring, there are three structural isomers. In general, OPSIN adds implicit brackets to attempt to yield a name with only one possible (non-degenerate) structural isomer, although perception of atom environments is not currently done to rigorously achieve this.

In general OPSIN implicitly brackets names, when two substituents are directly adjacent (e.g. no intervening locants/multipliers) to each other and the latter substituent has the `usableAsAJoiner` attribute. This attribute is generally present on substituents which possess only one substitutable hydrogen (e.g. formyl), all substitutable hydrogen on the same atom (e.g. sulfamoyl) or are a multi-radical accepting additive bonds (e.g. carbonyl).

OPSIN distinguishes between the case in which substituents are directly concatenated and the case in which they are separated by a hyphen; only the former are implicitly bracketed. This heuristic was found to be useful for interpreting chemical names generated by Lexichem.

When implicit brackets are added, locants could apply to the implicit bracket or the contents within it (Figure 3-132).



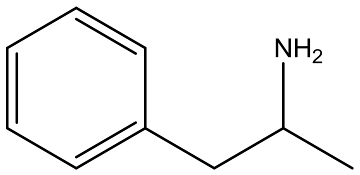
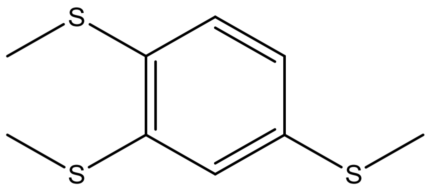


Figure 3-132 4-dimethylaminotoluene, interpreted as 4-(dimethylamino)toluene (left) but 2-aminopropylbenzene, interpreted as (2-aminopropyl)benzene (right)

Similarly multipliers could apply to the implicit bracket or to the contents within it (Figure 3-133).



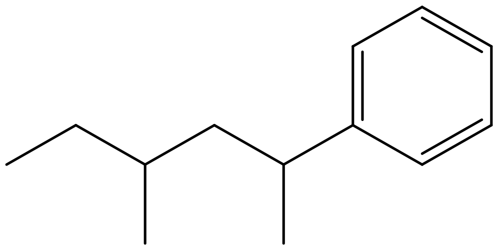
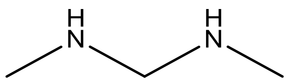


Figure 3-133 1,3,4-trimethylthiobenzene, interpreted as 1,3,4-tri(methylthio)benzene (left) but 1,3,4-trimethylbutylbenzene, interpreted as (1,3,4-trimethylbutyl)benzene (right)

Determining whether the locants and multipliers of the first substituent should be placed within the implicit bracket is heuristically determined by OPSIN considering whether the locant may apply to the other groups within the implicit bracket, the group itself or a group onto which it may be substituted. If a multiplier is a group multiplier e.g. 'bis' this is used as a hint that the multiplier describes multiplication of the implicit bracket (Figure 3-134).



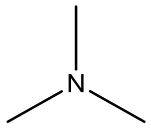


Figure 3-134 bismethylaminomethane, interpreted as bis(methylamino)methane (left) but dimethylaminomethane interpreted as (dimethylamino)methane (right)

3.2.15.2 Implicit spaces

Spaces are used in functional class nomenclature to separate the functional class of the compound from the substituent group. In most cases the absence of the space, with strict application of this rule, leads either to a name with a highly unlikely interpretation (Figure 3-135) or to a name with no interpretation e.g. ethylalcohol.

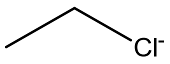


Figure 3-135 ethylchloride. Strictly this interpretation is not allowed as chloride possesses no substitutable hydrogen.

Hence, OPSIN does not enforce the presence of a space before a functional term and instead will treat such examples as if there were implicitly a space between the substituent and functional class term. This is done by having this construct of a substituent directly followed by a functional term actually present in the chemical grammar. The reason for this choice is that the parser is greedy and will consume as much input as it can interpret. A consequence of this is that if this construct were not in the grammar, chalcogen analogues of functional class terms would not be considered. This is because the chalcogen prefix would always be parsed by the grammar as a substituent instead of being considered as part of the functional term (Figure 3-136).

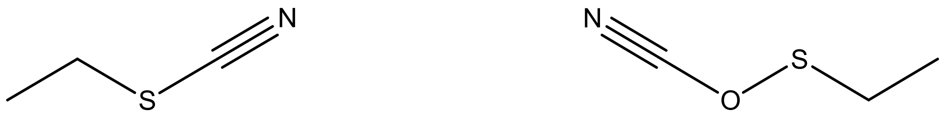


Figure 3-136 ethylthiocyanate or ethyl thiocyanate (left), ethylthio cyanate (right). For the space omitted name OPSIN generates parses for both interpretations before disambiguating in favour of the left-hand interpretation on the basis of having a longer functional term.

For esters disambiguation is more difficult as the space omitted form also produces a distinct chemically sensible interpretation (Figure 3-137).

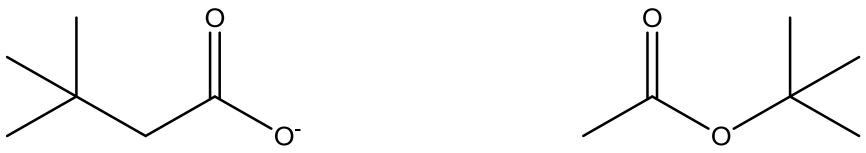


Figure 3-137 *tert*-butylacetate (left) and *tert*-butyl acetate (right)

Analysis of patents made it clear that strictly applying the IUPAC rules and treating such names as substituted anions was inappropriate.

OPSIN employs the following heuristics to distinguish between the cases where the omission of the space was intended and those in which an ester interpretation was intended. These criteria are applied before substituents are multiplied e.g. diethyl would be treated as one substituent.

- The first substituent in the name must have no locant and must be univalent. The multiplier (if present) in front of the substituent must not exceed the number of functional atoms present in the ‘ate’/ ‘ite’ group.

- If the parent group has exactly one substituent the ester interpretation is preferred if:
    - Substitution onto the 'ate'/ 'ite' group would lead to ambiguity. Ambiguity is determined through an analysis of the environments in which substitutable hydrogen are found using the same environment labelling as is employed during stereochemistry handling
    - It is prefixed with the multiplier 'mono'
    - The substituent is a straight chain alkyl chain followed by formate/methanoate/acetate/ethanoate. Such names produce an unambiguous anion interpretation but would not normally be named like this e.g. ethylethanoate would be called butanoate
- If the parent group has multiple substituents the ester interpretation is preferred if:
    - All substituents other than the first have locants (Figure 3-138)
    - The 'ate'/ 'ite' group has insufficient substitutable hydrogen atoms if and only if the substitution interpretation is assumed

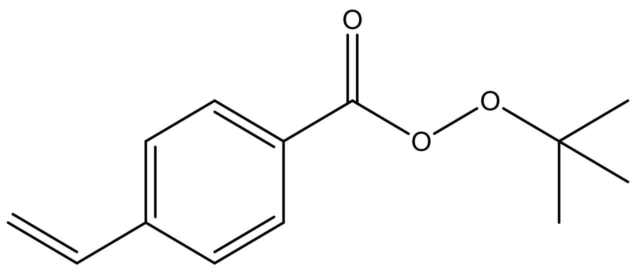


Figure 3-138 *tert*-butyl-4-vinylperbenzoate is interpreted as *tert*-butyl 4-vinylperbenzoate

Spaces may also be omitted in functional class names where the functional group is a divalent group and hence two substituents are expected. A long standing exception allows for one substituent to be omitted if both substituents are identical (Figure 3-139).

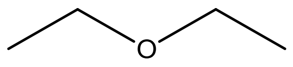


Figure 3-139 diethyl ether or ethyl ether (omitted substituent) or diethylether (omitted space) or ethylether (omitted substituent and space)

When two concatenated substituents are present before such functional groups OPSIN assumes that a space is omitted unless a locant is provided on the first substituent indicating that it connects to the second substituent.

3.2.16 Output formats

After a name has been interpreted, an OPSIN `Fragment` will have been generated that includes the molecule(s) described by the chemical name. This internal format may then be serialised to CML, SMILES or InChI.

3.2.16.1 CML

OPSIN's `Fragment`, `Atom`, `Bond`, `AtomParity` and `BondStereo` classes all contain a method to produce a CML serialisation which can be useful for debugging. The process of serialising a `Fragment` incorporates the results of serialising the constituent `Atom`s, `Bond`s, `AtomParity`s and `BondStereo`s. The CML serialisation differs from the other serialisations in that it also includes the locants associated with each atom (Figure 3-140).

```
<cml convention="conventions:molecular" xmlns="http://www.xml-
cml.org/schema" xmlns:conventions="http://www.xml-cml.org/convention/"
xmlns:cmlDict="http://www.xml-cml.org/dictionary/cml/"
xmlns:nameDict="http://www.xml-cml.org/dictionary/cml/name/">
    <molecule id="m1">
        <name dictRef="nameDict:unknown">propane</name>
        <atomArray>
            <atom id="a1" elementType="C">
                <label value="1" dictRef="cmlDict:locant"/>
                <label value="alpha" dictRef="cmlDict:locant"/>
            </atom>
            <atom id="a2" elementType="C">
                <label value="2" dictRef="cmlDict:locant"/>
                <label value="beta" dictRef="cmlDict:locant"/>
            </atom>
            <atom id="a3" elementType="C">
                <label value="3" dictRef="cmlDict:locant"/>
                <label value="gamma" dictRef="cmlDict:locant"/>
            </atom>
            <atom id="a4" elementType="H"/>
            <atom id="a5" elementType="H"/>
            <atom id="a6" elementType="H"/>
            <atom id="a7" elementType="H"/>
            <atom id="a8" elementType="H"/>
            <atom id="a9" elementType="H"/>
            <atom id="a10" elementType="H"/>
            <atom id="a11" elementType="H"/>
        </atomArray>
        <bondArray>
            <bond id="a1_a2" atomRefs2="a1 a2" order="S"/>
            <bond id="a2_a3" atomRefs2="a2 a3" order="S"/>
            <bond id="a1_a4" atomRefs2="a1 a4" order="S"/>
            <bond id="a1_a5" atomRefs2="a1 a5" order="S"/>
            <bond id="a1_a6" atomRefs2="a1 a6" order="S"/>
            <bond id="a2_a7" atomRefs2="a2 a7" order="S"/>
            <bond id="a2_a8" atomRefs2="a2 a8" order="S"/>
            <bond id="a3_a9" atomRefs2="a3 a9" order="S"/>
            <bond id="a3_a10" atomRefs2="a3 a10" order="S"/>
            <bond id="a3_a11" atomRefs2="a3 a11" order="S"/>
        </bondArray>
    </molecule>
</cml>
```

Figure 3-140 Example of CML output

3.2.16.2 SMILES

OPSIN includes a SMILES writer that can convert its internal format to SMILES. The SMILES writer includes support for everything that OPSIN's internal format can represent about the structure of a molecule, including stereochemistry. So as to produce shorter, more aesthetically pleasing SMILES hydrogens are supressed on all organic atoms except for nitrogens with double bond stereochemistry (Figure 3-141).

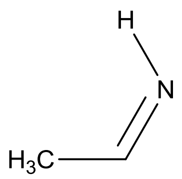


Figure 3-141 (Z)-ethanimine: C(/C)=N/[H] Note that without mentioning the hydrogen it is not possible to express this stereochemistry

SMILES descriptions for individual atoms and bonds can usually be generated in isolation from the rest of the molecule e.g. for an atom from its properties and hydrogen count. When stereochemistry is involved it is more complex as the serialisation is affected by the ordering of atoms within the SMILES string; hence the first step that the SMILES writer performs is a depth-first traversal of the molecule defining the order in which the atoms will be serialised. Double bond stereochemistry in conjugated systems is especially difficult as one must take into account the direction of slashes used for the previous double bonds as the same slash is used in the definition of the stereochemistry of both double bonds. OPSIN solves this by assigning consistent slash characters to all bonds to non-implicit atoms, which are adjacent to double bonds with defined stereochemistry before beginning writing of the SMILES string. In cases where neither group is an implicit hydrogen this leads to superfluous slashes but as they are not contradictory this is not incorrect (Figure 3-142).

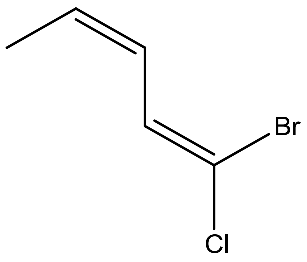


Figure 3-142 (1Z,3Z)-1-bromo-1-chloropenta-1,3-diene: Br\C(=C/C=C\C)\Cl

3.2.16.3 InChI

To create InChIs OPSIN employs the JNI-InChI library[128]. This allows the usage of InChI, a natively C library, through Java, on the majority of systems. The conversion from OPSIN's internal format to JNI-InChI's format is straightforward due to their near identical representations employed for describing stereochemistry. OPSIN can produce either standard InChIs or InChIs with fixed hydrogen layers. As IUPAC names generally specify a specific tautomer including the fixed hydrogen layer is preferred.

As JNI-InChI is a very large dependency compared to OPSIN's other dependencies, OPSIN is divided into two Maven modules. One of these contains OPSIN's core functionality, including CML and SMILES output, whilst the other solely adds the ability to do InChI serialisation.

3.3 Results and discussion

Evaluating chemical name to structure performance while theoretically simple is impeded by the difficulty of finding sufficiently large sets of accurately annotated name/structure pairs that are representative of the names of interest.

A study by Eller[129] found 26% of names in the analysed sample from the published literature to be formally unacceptable. When testing name to structure performance it is important to be able to know that conversion failures or unexpected name conversions are not just the result of the input name being incorrect. Eller also noted that machine generated names from the three pieces of name generation software tested (AutoNom 2000, ChemBioDraw and ACD/Name) produced formally incorrect names in only 1% of cases. For this reason all testing on the precision of chemical name to structure software has been performed on machine generated names. It should be noted that many major chemical drawing programs (e.g. ChemBioDraw, Marvin Sketch, Accelrys Draw, ACD ChemSketch) now incorporate structure to name algorithms, so finding machine generated chemical names in the literature is becoming increasingly common.

It is important to know that the findings on generated names are still applicable to chemical names "in the wild". One of the most commercially important applications of name to structure is locating chemical patents from the chemicals described within them. For this it is important to have high recall on the names used in such patents to describe exemplified compounds.

3.3.1 Methodology

3.3.1.1 Generated name test sets

The SMILES and InChIs for 30,000 randomly selected compounds were downloaded from PubChem, a database of more than 25 million small molecules. To randomly select the compounds, PubChem IDs were generated by random number generation in the range of valid IDs with removal of duplicates and revoked IDs until 30,000 valid IDs were generated.

The SMILES were then converted to names by ACD/Name 12.02, ChemBioDraw12, Lexichem 2.1.0 and Marvin 5.8.2. Due to an issue with ACD/Name's SMILES to name conversion including stereochemistry for double bonds, which did not have defined stereochemistry, an SDF generated by Lexichem from the SMILES was instead used as input to ACD/Name. InChIs were generated from these names by OPSIN 1.2.0, ChemBioDraw12, and Marvin 5.8.2. To give an indication of the difference in performance between OPSIN 1.2.0 and the version of OPSIN available at the commencement of this project, a version from November 2008 is included. As this version did not directly output InChIs, these were instead generated from the program's CML output using a simple Pybel[130] script as an interface to OpenBabel 2.3.1.

Determination of whether or not the InChIs were considered identical was made by comparison of the layers that are present in standard InChIs. Where the InChIs were not identical it was determined whether the layers that define the constitution of the molecule were identical. If they were, this was classed as a "Stereochemical Discrepancy", and, if they were different, this was classed as a "Constitutional Discrepancy".

As generated names are not expected to be correct in absolutely all cases a possible heuristic for detecting such cases is by looking at the consensus of name to structure solutions. For the cases where OPSIN failed to produce an identical InChI, the results of the other two name to structure programs was examined to determine whether either of them arrived at the correct InChI. If no solution could interpret a given name correctly this implies that the name may be suspect.

3.3.1.2 Chemical patents test set

USPTO patent applications that were filled in 2011 were downloaded from Google Patents[131]. The patents were filtered to just those containing organic chemistry (IPC code: C07). For each patent, `heading` elements were identified and their textual content passed to OSCAR4. Where OSCAR4 identified exactly one entity of type chemical, the surface of the entity, i.e. the name, was recorded. In the special case that the name (ignoring case) had been seen previously in the same or a previous patent, the name was not recorded. This filtering step helps with the problem that not all names present in headings will be exemplified compound names. A set of 248,846 names were extracted in this manner. Manual inspection indicated that the names are predominantly systematic in nature.

3.3.2 Data obtained

3.3.2.1 ACD/Name generated names

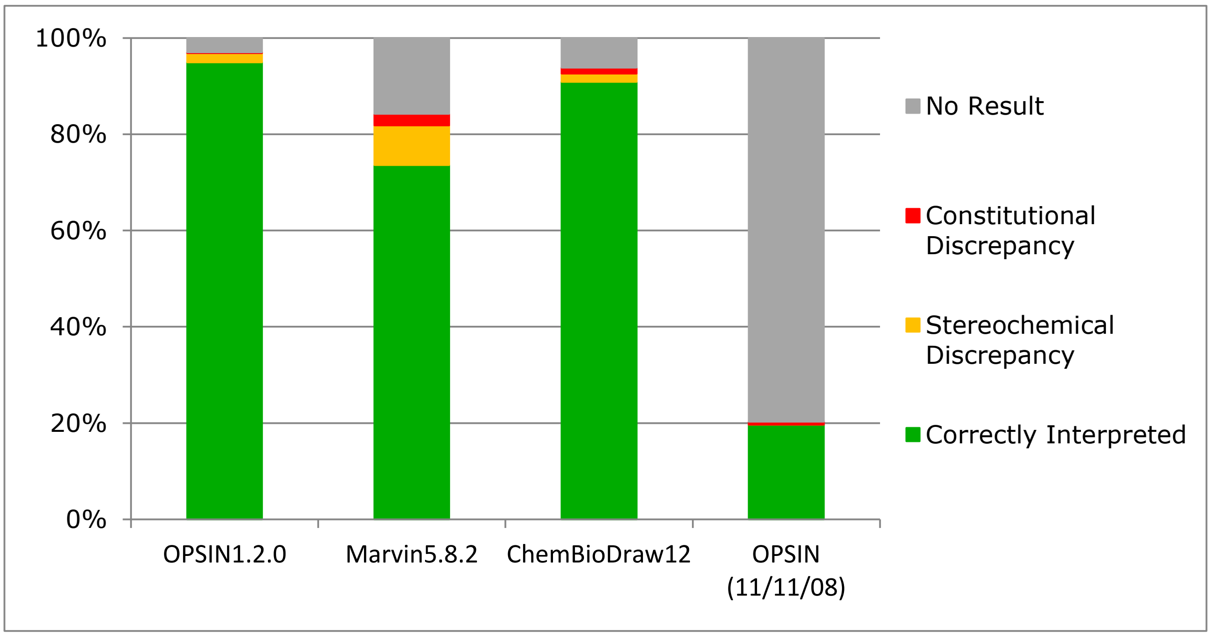


Figure 3-143 Comparison of performance on 29,718 ACD/Name 12.02 generated names

Names were tested as outputted by ACD/Name, with the exception that where present the string '(non-preferred name)' was removed from the end of names.

3.3.2.2 ChemBioDraw generated names

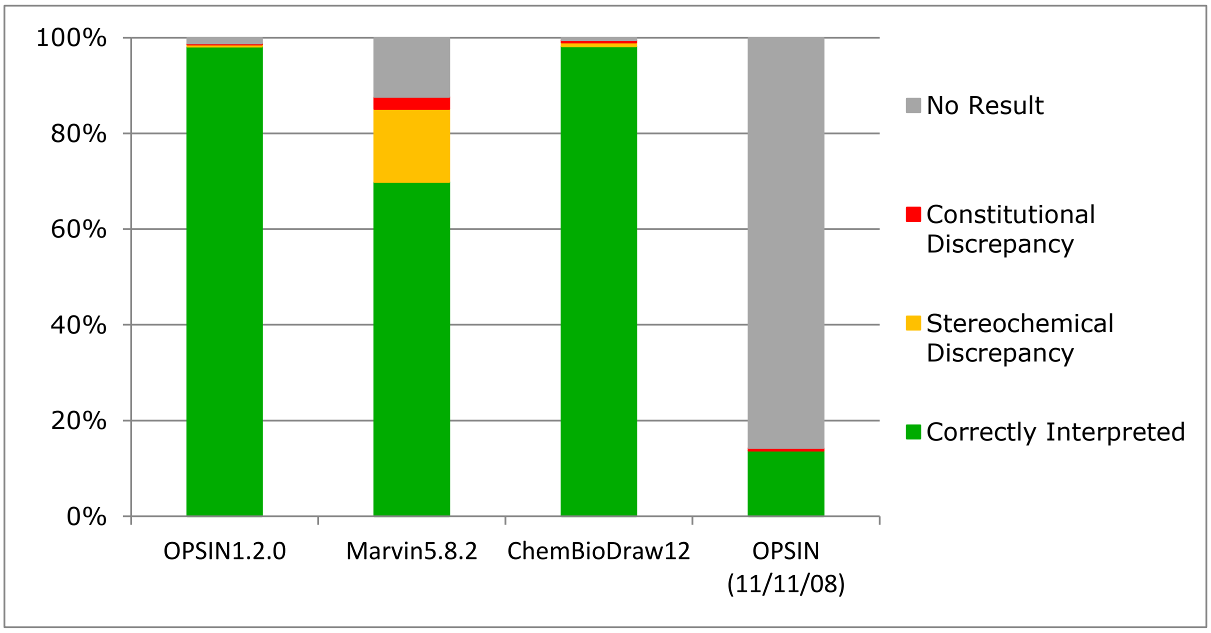


Figure 3-144 Comparison of performance on 29,414 ChemBioDraw12 generated names

Names were tested as outputted by ChemBioDraw.

3.3.2.3 Lexichem generated names

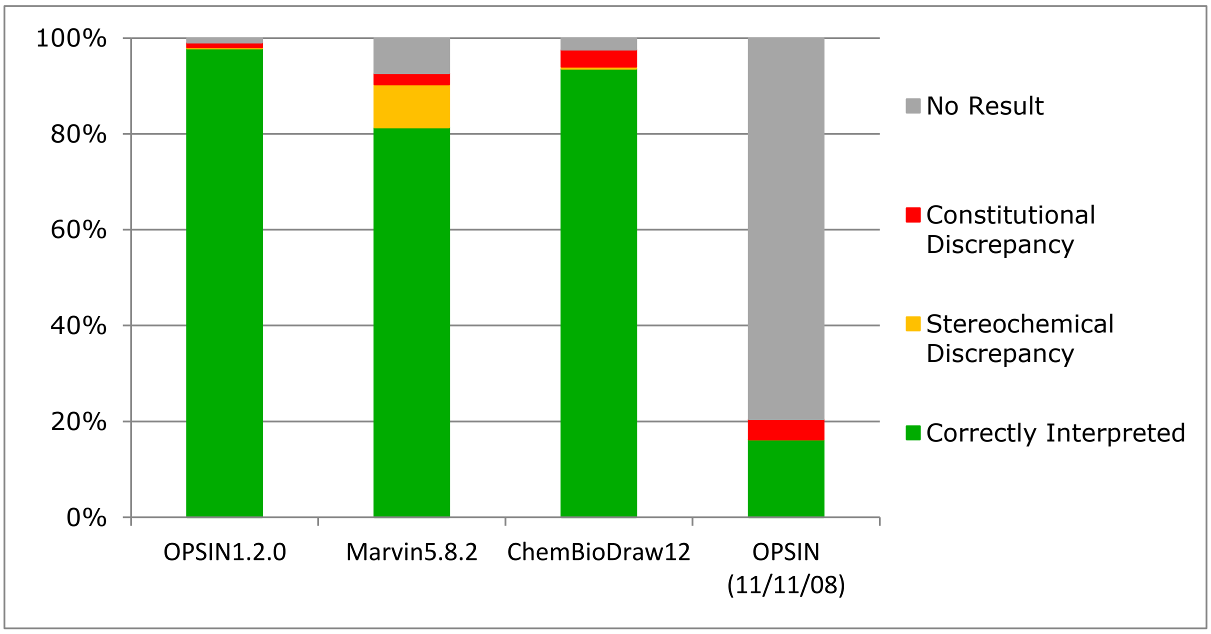


Figure 3-145 Comparison of performance on 29,301 Lexichem 2.1.0 generated names

Names were tested as outputted by Lexichem. On one exceptionally long systematic name Marvin failed to produce a result within 30 minutes necessitating the manual exclusion of that name.

3.3.2.4 Marvin generated names

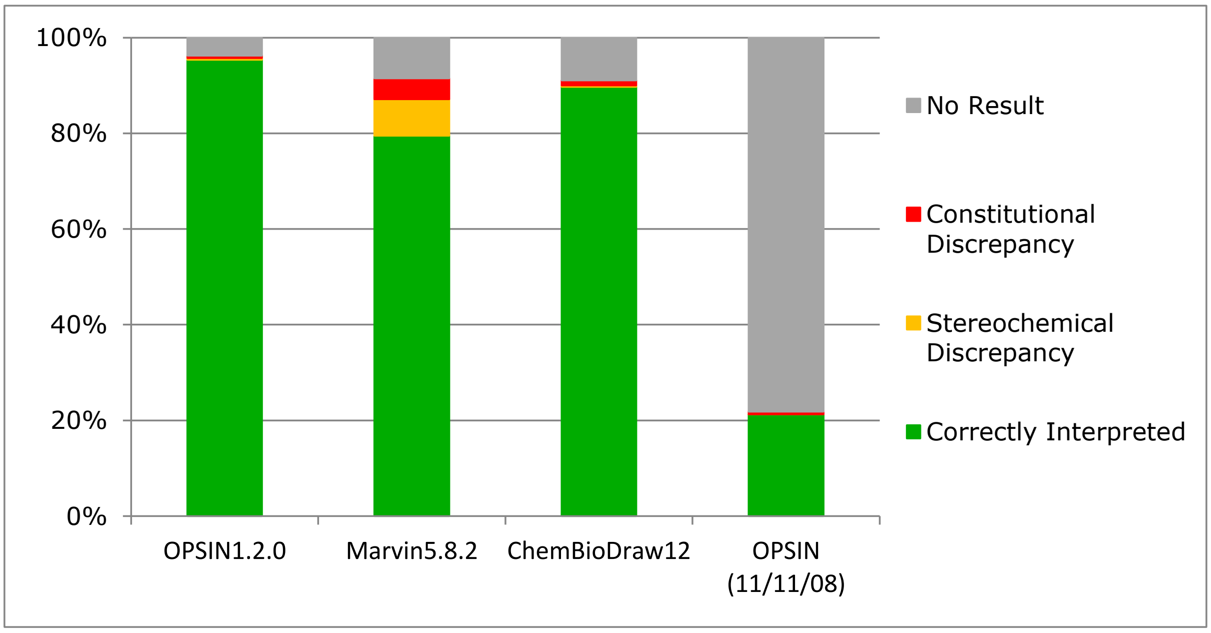


Figure 3-146 Comparison of performance on 29,961 Marvin 5.8.2 generated names

Names were tested as outputted by Marvin.

3.3.2.5 Compounds from headings in USPTO Patents

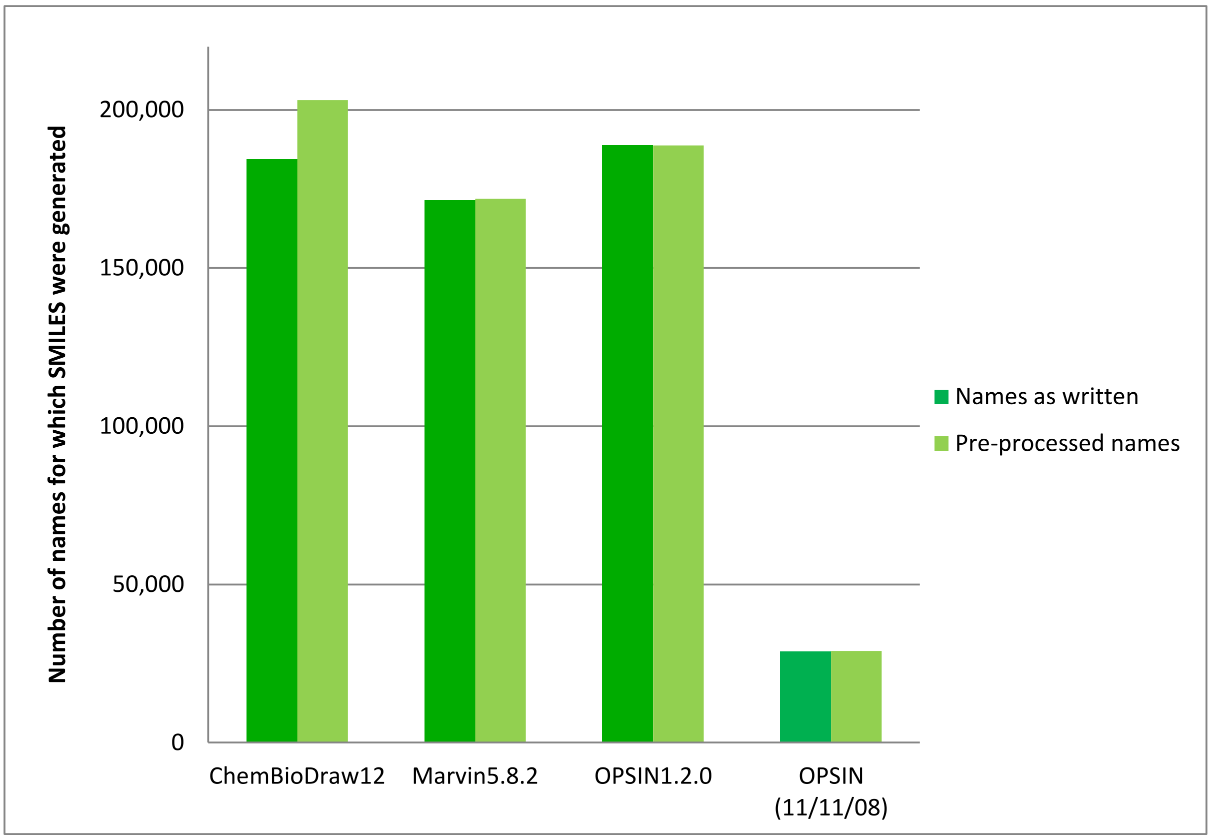


Figure 3-147 Comparison of recall on 248,846 names extracted from USPTO patents by OSCAR4. Pre-processed names were the result of passing the names through OPSIN's pre-processor.

The results in Figure 3-147 are intentionally not presented as a percentage of the size of the test set as at least 10% of the identified names are expected to be either false positives or contain insufficient information to generate a connection table. Unlike in the generated names, UTF-8 characters beyond the ASCII subset were frequently encountered e.g. Greek letters (α) and primes (′). A significant percentage of ChemBioDraw's failures were purely due to the use of these characters hence the names were passed through OPSIN pre-processor (Section 3.2.3) to allow assessment of the level of nomenclature coverage rather than of ChemBioDraw12's ability to recognise non-ASCII characters. For names containing characters unrecognised by OPSIN's pre-processor the original name was retained.

3.3.3 Discussion

The results show that OPSIN has consistently high levels of recall (96.2% - 99.0%) and precision (97.9%-99.3%) across all the sets of generated names. While precision as stated is high, many of the failures maybe expected to be the result of the names being incorrect. Table 3-15 shows that the majority of the names that OPSIN incorrectly interpreted, were also incorrectly interpreted by Marvin and/or ChemBioDraw. In the paper on OPSIN[72], an older version of OPSIN, on different sets of generated names in which incorrect and ambiguous names were identified and excluded from the precision calculations, was able to achieve precision in excess of 99.8%. Different sets of names were used than those in the paper, as in the course of creation of the paper the author manually checked all names that produced discrepant results to determine whether the fault lay with the name. This analysis allowed, subsequently to the paper, for the majority of the genuine errors made by OPSIN to be corrected, but as a result these sets cannot be considered unseen test sets.

	Sets of names			
	ACDName12.02	**ChemBioDraw12**	**Lexichem2.1.0**	**Marvin5.8.2**
Can be converted correctly by a solution	4.1%	45.5%	7.7%	17.0%
Can't be correctly converted but can be incorrectly converted	71.8%	51.8%	83.3%	68.4%
Can't be converted by either solution	24.1%	2.6%	9.0%	14.6%

Table 3-15 Analysis of how the union of ChemBioDraw and Marvin handled the names that OPSIN 1.2.0 produced discrepant results on.

OPSIN's names showed a lower level of agreement with the starting structures when using names generated by ACD/Name (Figure 3-143), as compared to when using names generated by the other software. This arises from the use, by ACD/Name, of amino acid names without ᴅ/ʟ prefixes to describe amino acid components of the structure without defined stereochemistry. The IUPAC recommendations[118(Rule 3AA-3.3)] state that the meaning of an amino acid name without the prefix depends on the context e.g. if the amino acid is known to come from a natural source it may be assumed to be ʟ whilst if it known to be synthetic it may be assumed to indicate a racemate. OPSIN, and indeed the other name to structure solutions tested, assumes the ʟ configuration in all cases leading to apparent discrepancies in results.

In OPSIN's publication[72] it was found that ChemBioDraw was sensitive to the representation of superscripts and Greek letters used by other structure to name packages e.g. $a for alpha or ^ to indicate superscripts. Pre-processing the chemical names to use representations understood by

ChemBioDraw may slightly improve its performance on the ACD/Name, Lexichem and Marvin generated names.

Across all four sets of names Marvin can be seen to have generated stereochemically discrepant results in a large percentage of cases. This appeared to a large extent to be caused by difficulties in its algorithm correctly identifying the stereocentre to which indicated stereochemistry should be applied. For example '(S)-bromo(chloro)fluoromethane' was interpreted without stereochemistry whereas 'bromo(chloro)fluoro-(S)-methane' was correctly interpreted.

The results on the names extracted from patents (Figure 3-147) also showed excellent performance from OPSIN, giving significantly higher recall than Marvin. Comparison to ChemBioDraw is more difficult as dependent on whether or not the names are pre-processed OPSIN either had slightly higher or slight lower recall. Correspondence with a ChemBioDraw developer indicated that the lack of support for non-ASCII Unicode characters was a bug that would be corrected in the next version.

The difference between OPSIN's current performance and the level potentially achievable with ChemBioDraw is likely to be explained by OPSIN's lack of support for some areas of carbohydrate nomenclature as well as ChemBioDraw's greater leniency in handling names that do not conform to codified nomenclature practices.

3.4 Implementations

3.4.1 Java library

OPSIN's main mode of distribution is as a Java library typically including both the core and InChI modules. The API has been designed to offer convenience methods for the most commonly required capabilities in conjunction with more advanced configurability. The methods in the public API of `NameToStructure` are listed below:

Method	Output
`parseToCML(String name)`	`nu.xom.Element`
`parseToSmiles(String name)`	`String`
`parseChemicalName(String name)`	`OpsinResult`
`parseChemicalName(String name, NameToStructureConfig n2sConfig)`	`OpsinResult`
`getOpsinParser()`	`ParseRules`

The `parseToCML` and `parseToSmiles` are convenience methods and allow the direct conversion of a chemical name to the relevant format e.g. a CML document and a SMILES string respectively, using the program's default options. A CML document is returned as a XOM Element object allowing in-memory manipulation or trivial serialisation to XML.

Alternatively the output may be an `OpsinResult`. This contains whether name interpretation was successful, the error message that was returned (if applicable) and the name that was interpreted. An `OpsinResult` may be lazily serialised to either CML or SMILES using the class' methods.

If greater configurability is desired, a `NameToStructureConfig` object can be provided that allows configuration of OPSIN's options (Table 3-16).

Option	Explanation	Default value
allowRadicals	Should names that formally describe radials be accepted e.g. ethyl	false
detailedFailureAnalysis	If a chemical name is uninterpretable should OPSIN parse it from right to left to attempt to generate a more informative error message	false

Table 3-16 OPSIN's configurable options

The `ParseRules` object returned by getOpsinParser allows the parsing of words using OPSIN's grammar. This functionality is employed extensively by the OPSIN Document Extractor (Section 3.4.4) but is not known to be employed elsewhere. Note that generally only a single word may be parsed at a time e.g. 'ethyl ethanoate' will not be fully parsable but 'ethyl' or 'ethanoate' are parsable.

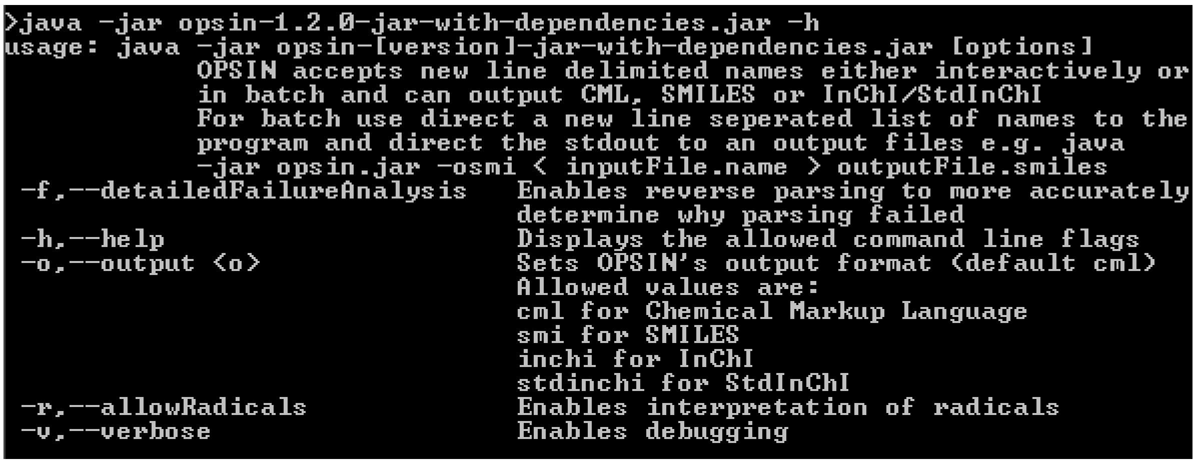
If one wishes to debug OPSIN's behaviour an end user may achieve this by setting the Log4J log level to either `debug` or `trace` depending on the level of detail required.

Library functions for InChI generation reside in the `NameToInchi` class in the InChI module. Functions are available for the generation of an InChI with fixed-H layer or a StdInChI from an `OpsinResult`. Convenience methods are also available to go directly from a name to either form of InChI.

The library is available either from the project's download page on BitBucket[132] or from the Maven central repository.

3.4.2 Command-line interface

When OPSIN is distributed in library form as an executable jar file, execution yields a command line interface. Flags are available to set all of OPSIN's configurable options, the desired output format and verbosity (Figure 3-148). Verbose output corresponds to a Log4J log level of `debug`. The same command-line is employed regardless of whether the InChI module is included, hence to avoid the command-line interface depending on the InChI module, reflection is used to check for the presence of the InChI functionality on the classpath. The command-line interface may be used to perform batch processing by piping in a file of chemical names and directing the output to an appropriate output file.



```
>java -jar opsin-1.2.0-jar-with-dependencies.jar -h
usage: java -jar opsin-[version]-jar-with-dependencies.jar [options]
            OPSIN accepts new line delimited names either interactively or
            in batch and can output CML, SMILES or InChI/StdInChI
            For batch use direct a new line seperated list of names to the
            program and direct the stdout to an output files e.g. java
            -jar opsin.jar -osmi < inputFile.name > outputFile.smiles
 -f,--detailedFailureAnalysis   Enables reverse parsing to more accurately
                                determine why parsing failed
 -h,--help                      Displays the allowed command line flags
 -o,--output <o>                Sets OPSIN's output format (default cml)
                                Allowed values are:
                                cml for Chemical Markup Language
                                smi for SMILES
                                inchi for InChI
                                stdinchi for StdInChI
 -r,--allowRadicals             Enables interpretation of radicals
 -v,--verbose                   Enables debugging
```

Figure 3-148 Screenshot of OPSIN command line help dialog showing available flags

3.4.3 OPSIN web service

The OPSIN web service[133] provides access to OPSIN's functionality to convert names to CML, SMILES and InChI via a convenient web interface. Additionally the web interface can generate depictions using the Indigo toolkit[55]. The Indigo toolkit is also used to enrich the CML with generated 2D coordinates.

Requests to the web interface may be either done using a browser by entering a chemical name at opsin.ch.cam.ac.uk or programmatically by sending requests to opsin.ch.cam.ac.uk/opsin. Requests may be made using content negotiation or by adding a suitable file extension to the request (Table 3-17).

Request type	Internet media type	File extension
CML	chemical/x-cml	.cml
CML without 2d coordinates	n/a*	.no2d.cml
SMILES	chemical/x-daylight-smiles	.smi
InChI	chemical/x-inchi	.inchi
Depiction	image/png	.png

Table 3-17 Request types supported by the OPSIN web service. *chemical/x-no2d-cml is accepted but is not a recognised internet mime type

The web service is employed by the Chemistry Add-in for Word[134], a joint development between the Unilever Centre and Microsoft, as a means of converting chemical names to chemical objects.

The web service's logs were analysed over a one week period in early December 2011 showing requests from 171 unique IP addresses. Usage patterns varied from single names all the way through to automated requests for 1000s of names. Analysis of failing web service requests has revealed that the vast majority of failures have been caused by unrecognised trivial names (e.g. drug names), spelling mistakes, non-English chemical names and non-names (e.g. SMILEs, molecular formulae etc.). The few genuine failings have proven of some use in finding "bugs" and areas of unsupported nomenclature.

When a failure is encountered the web service employs OPSIN's reverse parsing to attempt to identify the exact part of a name that is uninterpretable in the error response. Users of the service have reported this to be useful in identifying and correcting errors in chemical names[135].

3.4.4 OPSIN Document Extractor

The OPSIN Document Extractor[136] attempts to find all sequences of words that are parsable by OPSIN. This is assumed to indicate that, with a high degree of confidence, the identified strings are chemical names. The program works as follows on a string of text:

- Whitespace tokenisation to form an array of words. The character indices of these words in the original string are recorded.
- OPSIN's pre-processor is employed to generate an array of normalised words which will be operated on henceforth.

- Identification of stop words e.g. ‘on’, ‘one’, ‘at’. These are English words that can also be the ending of chemical names (often German chemical names) and should be prevented from forming chemical names.
- The words are parsed by OPSIN in pairs. Depending on whether or not OPSIN believes a word to be interpretable on its own, the program may add one or both words to a buffer of successfully parsed name fragments e.g. ‘ethyl benzene’ would be consumed in two cycles but ‘benzoic acid’ or ‘chloral hydrate’ would be consumed as one.
- If a pair of words is partially interpretable and the point of failure does not occur at a word boundary, spaces are removed until either no improvement in the length of name that is interpretable is noticed or the chemical name ends at a word boundary.
- As OPSIN knows the role of chemical words and whether they are valid on their own, intelligent choices can be made as to whether space removal should be attempted. For example ‘benzene sulfonamide’ should be ‘benzenesulfonamide’ but ‘pyridine acetic acid’ should be interpreted as is, rather than treating the acetic acid as a conjunctive substituent of the pyridine ring.
- Punctuation at the end of a chemical name, or a bracketed section immediately following a chemical name is ignored and indicates the chemical name is complete. A chemical name is also indicated as being complete if a subsequent word cannot be interpreted as being chemical or the end of the array of words is reached.
- Identified chemical names are classified as “complete”, “part”, “family” or “polymer”. “part” names are names classified by OPSIN as substituents. “family” names are classed by OPSIN as functional terms or are names that end in an ‘s’ which could not be interpreted by OPSIN. “polymer” names start with the functional term ‘poly’ or ‘oligo’.
- An unbalanced opening bracket at the start of a chemical name, or an unbalanced closing bracket at the end of a chemical name, is removed. Balanced brackets surrounding a chemical name are removed. A terminal ‘-’ or ‘,’ is removed e.g. ‘ethyl-’ is recognised as ‘ethyl’
- The output is a list of identified chemical names which can be queried for the normalised chemical name, the raw text, the chemical name classification, the start and end character indices within the original string and the start and end positions within the array of words.

As the program knows whether punctuation is valid as part of a chemical name, individual chemical names may still be extracted from lists of chemical names even in the presence of erroneous whitespace (Table 3-18).

Input: 'indane, 1,2, 3,4- tetrahydroquinoline, 3, 4-dihydro-2H-1, 4-benzoxazine, 1,5-naphthyridine, 1, 8- naphthyridine'

Identified chemical name	Text value
indane	indane
3,4-tetrahydroquinoline	3,4- tetrahydroquinoline
3,4-dihydro-2H-1,4-benzoxazine	3, 4-dihydro-2H-1, 4-benzoxazine
1,5-naphthyridine	1,5-naphthyridine
8-naphthyridine	8- naphthyridine

Table 3-18 Output from OPSIN Document Extractor on a list of chemical names containing erroneous whitespace

The OPSIN Document Extractor is utilised as a tagger for use with ChemicalTagger (as described in Section 4.4.5.6) and as an aid in name type assignment (as described in Section 4.5.1.4). It should be emphasised that, whilst the approach taken by the OPSIN Document Extractor is rather brute force in nature, it is still typically an order of magnitude faster than performing entity recognition with OSCAR4. Hence, using the OPSIN Document Extractor as a complement to OSCAR4, as is done in the work on reaction extraction described in Chapter 4 of this thesis, may be done with minimal effect on performance.

3.5 Areas for future work

3.5.1 Vocabulary

Chemical name to structure is impossible when vocabulary unrecognised by the program is encountered. Hence, an obvious improvement to OPSIN is the addition of more terms to its vocabulary. This is especially important in the area of natural products, for which even the majority of IUPAC recommended alkaloid and terpenoid trivial names, listed in the natural product recommendation[127 (Appendix)], are yet to be added. A site offering an extensive list of trivial names with corresponding systematic names and Japanese names was identified but unfortunately time was insufficient to fully add more than just the acyclic trivial names[137]. Addition of trivial names is complicated by the question of what category in the grammar to add the name to e.g. can the name have suffixes, if so which suffixes? Other concerns are getting the numbering of the compound correct when the compound has an accepted numbering system, and, especially in the case of natural products, making sure the structure has correct stereochemistry. The stereochemistry problem is made more difficult by the proliferation in databases of structures with slightly different stereochemistry that have become erroneously associated with the same trivial name[138].

The addition of vocabulary can also assist with the problem of trivial names that are composed of understood morphemes which are then deconstructed into their apparent morphemes. This poses a problem as the apparent morphemes may not precisely describe the structure or may be wholly misleading (Figure 3-149). The addition of appropriate trivial names allows the systematic interpretation to be overridden.

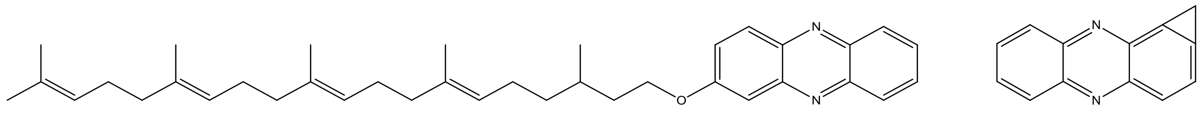


Figure 3-149 Methanophenazine (left) and a systematic interpretation of it (right)

3.5.2 Carbohydrate nomenclature

OPSIN currently possesses support for carbohydrates with the suffix 'ose' optionally infixed with a ring size specifier to give suffixes such as 'pyranose'. Adding support for more suffixes especially those that allow groups named by carbohydrate nomenclature to be used as substituents would yield a significant improvement in recall. Adding further suffix support is not entirely trivial due to only some suffixes being locantable and some suffixes applying to multiple atoms but extension of OPSIN's existing mechanisms for handling similar cases should be sufficient.

Oligo saccharide nomenclature which employs arrows to indicate the linkage between saccharides could be relatively trivially support by internal conversion to normal locants e.g.

α-ᴅ-glucopyranosyl-(1→4)-β-ᴅ-glucopyranose could be internally converted to:

O^4-(α-ᴅ-glucopyranosyl)-β-ᴅ-glucopyranose

3.5.3 Inorganic nomenclature

Inorganic nomenclature is mostly unsupported by OPSIN. The reason for this stems from two problems:

- Datively bonded substituents are often named by the name of the group e.g. 'amine', 'pyridine' etc. To classify these as substituents rather than parent groups one needs to know that they are followed by an inorganic parent. Treating them as parent groups will not work as they may be preceded by ligands that are expressed as substituents e.g. ‘chloro’ which will be referring to the metal rather than the datively bonded substituent e.g. ‘dichlorodipyridine platinum(II)’.

- OPSIN’s current internal format does not have good support for representing dative bonds and other non-covalent interactions e.g. the interaction between the π electrons and the iron atom in ferrocene. This same is also true to varying extents of the formats to which OPSIN writes.

The issues of representing inorganics are dealt with by Clark[139] who recommended the introduction of a zero-order bond to the commonly used MDL chemical table file formats. This would mostly solve the problems of representation although the exact semantics of the interactions would be lost. It would also be insufficient to correctly represent systems with three centre two electron covalent bonds e.g. diborane. In these systems representing one bond as a single bond and the other as a zero order bond artificially introduces asymmetry.

Until support for file formats that allow better specification of inorganics becomes more widespread improving support for inorganic nomenclature is unlikely to benefit most cheminformatics applications.

3.5.4 Stereochemistry

Many forms of stereochemistry remain unsupported including *endo*, *exo*, *syn*, *anti*, r, s, e, z and α/β stereochemistry on arbitrary ring systems. Adding rigorous detection for pseudo-asymmetric centres would be an important precursor to further improving stereochemistry handling by OPSIN. Currently only a limited subset of the pseudo-asymmetric centres are detected. For r, s, e and z, extension of OPSIN’s implementation of the CIP rules would also be required.

3.5.5 Nomenclature variants

Even if OPSIN were to support all codified nomenclature, there still remains the long-tail of nomenclature variants that appear, both intentionally and unintentionally, in “real world” use. This is an unbounded problem as the chemistry community can always think of new ways to construct chemical names that will be unexpected to a computer program. In this respect an approach like that taken by Name=Struct may be advantageous over a grammar-based approach although this must be weighed against the increase in erroneous structure conversions which will inevitably occur when odd nomenclature is encountered.

3.5.6 Detection and handling of ambiguous names

OPSIN has not been designed to detect ambiguous chemical names and hence introducing such functionality would involve substantial changes especially if alternative structure interpretations were to be enumerated. The codebase now includes code for atom environment detection and indeed this is actually employed for detection of ambiguity in the very specific case of determining whether an ester interpretation is desired (Section 3.2.15.2). The application of this technique to substitutive nomenclature operations would be sufficient to detect a significant number of cases of ambiguity although structural ambiguity may be introduced by many other nomenclature operations for which a similar analysis would also need to be performed.

A significant complicating factor is determining whether a name that is formally ambiguous should be considered unambiguous by convention. In the example of p-aminomethylbenzene-sulfonamide (Figure 3-150) the name is formally ambiguous as 'aminomethyl' is not bracketed. In practice there is only one likely interpretation, as the name would otherwise contain a methyl group that could be placed at multiple positions on the benzene ring whilst still being consistent with the name.

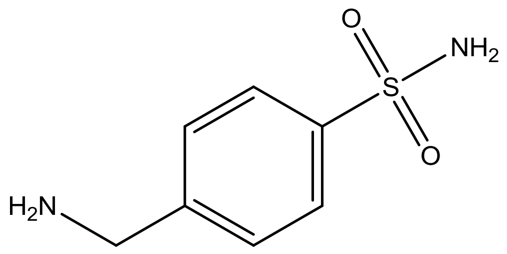


Figure 3-150 p-aminomethylbenzene-sulfonamide

3.5.7 Detection of typographical errors

The problem of detecting and correcting typographical mistakes was investigated, during the course of this project, yielding a proof of concept system. This worked by parsing the chemical name to the point at which no further tokens could be found then determining if one operation could change the chemical name such that it would match one of the tokens present in the allowed token classes. These operations were substitution, insertion, deletion and transposition, which have been found to account for 80% of typographical errors[140]. Due to the existence of morphemes in the same token class that differ by only a single letter e.g. 'amino', 'imino', only substitutions between letters that were adjacent on a US keyboard were allowed. Where multiple possible suggestions were possible a heuristic, that chose the token that appeared more often in a training corpus, was invoked.

This work yielded promising results. The only significant drawback being that, if the typographic mistake was before the point in the name at which parsing failed, the mistake could not be corrected as backtracking through the grammar's finite state machine had not been implemented. This work was not taken forward primarily due to concerns that the results would still not be accurate enough for automated use in text mining. Nonetheless adding such functionality to applications such as the OPSIN web service could be useful.

3.5.8 Foreign language support

OPSIN includes some support for German names purely by making the terminal 'e' at the end of many chemical names optional. An experiment with adding further German vocabulary flagged up an ambiguity that would be introduced in the parsing of 'chloro-'. With the addition of the German 'chlor' this could then also be parsed as [chlor][o-] where 'o-' is an ortho locant, hence the German specific vocabulary is currently not enabled to avoid introducing ambiguity into unambiguous English names.

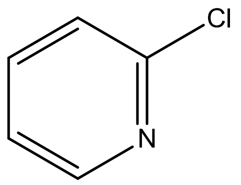


Figure 3-151 English: 2-Chloropyridine; German: 2-Chlorpyridin (unsupported due to 'chlor' not being in vocabulary, 'pyridin' is allowed as OPSIN considers the 'e' optional)

A small proof of concept attempt was made to support Chinese chemical names indicating that for Chinese many English morphemes could be simply replaced with Chinese characters due to the underlying grammar being mostly the same. In some areas though the syntax was found not be identical e.g. alkanes are ordered by hundreds, then tens then units whilst in English IUPAC names the ordering is reversed. Modifying OPSIN's grammar or enumerating such systematic constructions are not especially elegant solutions. In languages such as French the ordering of words may be different e.g. 'acide formique' which poses further problems.

Sayle[141] described a method whereby names in foreign languages could be translated from, and to, English through a mixture of word order rearrangement and morpheme string substitutions (some of which were context sensitive). This is expected to be the more elegant solution although the inherent disambiguation that a grammar-based system like OPSIN provides may give more elegant solutions in cases where context sensitive substitutions are required.

3.6 Conclusions

This project has resulted in the creation of a fast, precise and extensible chemical name to structure interpretation algorithm. By employing a strict grammar, OPSIN can elegantly fail on chemical names that include nomenclature that is not yet supported. OPSIN is known to be employed by AMBIT[142], Cinfony[143], the National Cancer Institute's Chemical Identifier Resolver[144], Bioclipse[145], LICSS[146], OCMiner[147], Digital Science's SureChem[36] and at the International Union of Crystallography[148], Dupont[149], AstraZeneca[150] and IBM[151]. This wide range of users encompasses text mining efforts and more general applications in which name to structure can be a time saving mechanism.

Newly synthesised compounds, and to a lesser extent reagents, are often referred to by systematic names; hence the success of the work described in the next chapter on extracting reactions from patents was only possible due to the high recall and precision afforded by OPSIN.

Chapter 4 Extraction of Chemical Reactions from the Patent Literature

4.1 Introduction

Reaction databases are primarily employed by synthetic chemists to find ways to perform a particular synthesis or synthetic step. They may already know the reaction they are interested in performing and hence want to investigate the conditions employed in successful instances of the reaction. Alternatively they may be interested in identifying reactions that would, or have the potential to, lead to the formation of a particular moiety.

The largest reaction databases are the commercial CASREACT[152,153] and Reaxys[154,155] databases each containing in excess of 30 million reactions. There are many smaller commercial databases e.g. SPRESI[156,157], Current Chemical Reactions[158], Science of Synthesis[159] and SORD[160] (free to academics).

As compared to structural databases, where freely accessible databases like ChemSpider and PubChem rival the size of the leading commercial databases, freely accessible reaction databases are currently comparatively small in size. Such databases include the journal Organic Syntheses[161] and WebReactions[162] which uses the ChemReact database, a subset of the SPRESI database.

Reaction databases are generally populated by manual abstraction of reactions from the chemical literature. This is highly time consuming work and hence, due to the associated costs, large scale abstraction is only practical for the largest commercial databases. Automated techniques for reaction extraction have the potential, where primary literature is readily text minable as is the case for patents, to allow the creation of large reaction databases with extremely low costs. Such techniques may also find utility in expediting work to manually extract reactions from the literature by providing crude extracted reactions which could then be tweaked by human curators.

This chapter describes the development of an open source system for the automatic extraction of reactions from the chemical literature especially patents. The developed system was presented at the spring 2012 ACS conference[163]. The description of the system unless specified otherwise refers to v1.0 of the developed software.

4.2 Previous attempts at text mining chemical reactions

4.2.1 Chemical Abstracts Service

Blower et al.[164–167] from the Chemical Abstracts Service published a series of papers spanning the period 1983-1990 in which they discuss automated methods for extracting reactions from the American Chemical Society's Journal of Organic Chemistry. Their system modelled experimental sections as being formed of a heading, a synthesis, a workup and a characterisation section with only one resultant product. This model was found to describe over half the experimental sections they encountered.

The original system[165,166] worked by tokenising on common delimiters with appropriate rules to differentiate between hyphens within chemical names and within other words. Words were assigned part of speech tags or as chemical words by a mixture of dictionary lookup and looking at the stem and suffix of words. A rule based system was used to disambiguate in cases where context is necessary to accurately determine the part of speech. Assigning roles to reagents was partially achieved using a "word expert" system that would be able to use surrounding words e.g. 'in' or 'under' to assign a likely role to each reagent. The words preceding and following each reagent were then scanned for quantities which were associated with the appropriate reagent.

The discourse (heading/synthesis/workup/characterisation) was determined by a set of criteria. The heading was determined by the absence of a verb in the words that made it up. The synthesis section was not identified directly but instead assumed to be the content between the heading and workup section. The workup section was identified by the presence of words from a list a list of common operations performed at this stage e.g. crystallise, wash etc. The characterisation section was identified by the presence of acronyms commonly associated with characterisation e.g. mp, m/e etc.

The 1990 paper[167] described a more refined approach taking inspiration from the previously described system. Partial parsing of sentences is achieved using Augmented Transition Network parsing; a parsing method based on the use of a finite state automaton that can accept words as transitions and that allows nondeterministic transitions hence allowing recognition of content-free languages. The parsing attempts to identify substance information, references to procedures, time/temperature data, verb phrase and characterisation data. The system included some support for general procedures (where a template for a reaction is given, optionally followed by specific instances of the reaction in which not all reagents are specified), analogous syntheses (where a compound is obtained in the same way as a previously described procedure) and parallel synthesis (where the synthesis of several analogous compounds is given at once). The program was originally intended to assist in abstracting for CASREACT. However, it was not deemed sufficiently accurate, being able to produce “usable” results from 80-90% of simple synthesis paragraphs and only 60-70% of the more complex cases.

While the reaction extraction system developed as part of this project is more sophisticated in terms of chemical entity recognition and chemical entity resolution (something not even attempted by the program) the range of experimental paragraph types supported still go beyond the scope of what has been developed for this project.

4.2.2 University of Cambridge

Jessop et al[168] developed a system, coined PatentEye, which was employed to extract reactions from EPO patents. Verification of the structures of reaction products was attempted through comparison of the result from a chemical name to structure algorithm (OPSIN), with those obtained from an image to structure algorithm (OSRA). The structure could also be checked for consistency with extracted NMR and mass spectra. With the version of OSRA utilised, the results from image to structure conversion were insufficiently precise to allow verification of the products with only 34% of a set of 200 images being converted exactly to the human reproduced structures. Although the majority of NMR spectra could be successfully extracted, exactly predicting the peaks in an NMR spectrum is a complicated process. This made it difficult to be sure that a spectrum was or wasn’t consistent with a given product structure. While a case was firmly made for the utility of capturing spectral information the case for using this information or image to structure results to verify product information was less clear considering the relatively high accuracy of chemical name to structure software, hence this was not pursued in the current work.

The workflow: sectioning of a document into experimental section, derivation of chemical structures using OPSIN/OSCAR/OSRA and application of ChemicalTagger to identify reagents and assign them roles and quantities, is broadly similar to the work described in this chapter. While with the exception of the paragraph classifier (Section 4.4.4) there is no code in common between the projects this project can be considered a spiritual successor. The most significant difference between these projects is that the current work puts a far greater emphasis on the extracted structures. The structures are used to assist in role assignment and facilitate the atom-mapping step that checks that a reaction is feasible.

4.2.3 University of Toronto

Since 2001, ChemDraw binary CDX files and MDL Molfiles are available with USPTO patents The CDX files are submitted by the patent applicant and hence offer another source of information from which reactions may be extracted. Work at the University of Toronto has culminated in the production of the SCRIPDB database of structures derived from these CDX files[169]. As of the end of 2010, SCRIPDB contained 10,840,646 molecule instances (molecules were de-duplicated on a per patent basis). CDX files may also contain indication that compounds are involved in a reaction step and the relationship between the reactions steps. 341,764 reaction steps were identified up till the end of 2010. Correspondence with the author indicated that the number of reactions present in the CDX files may be potentially up to double these values due to the CDX file in many cases having the appropriate graphical elements (e.g. reaction arrow) but lacking the semantic indication that a reaction is described.

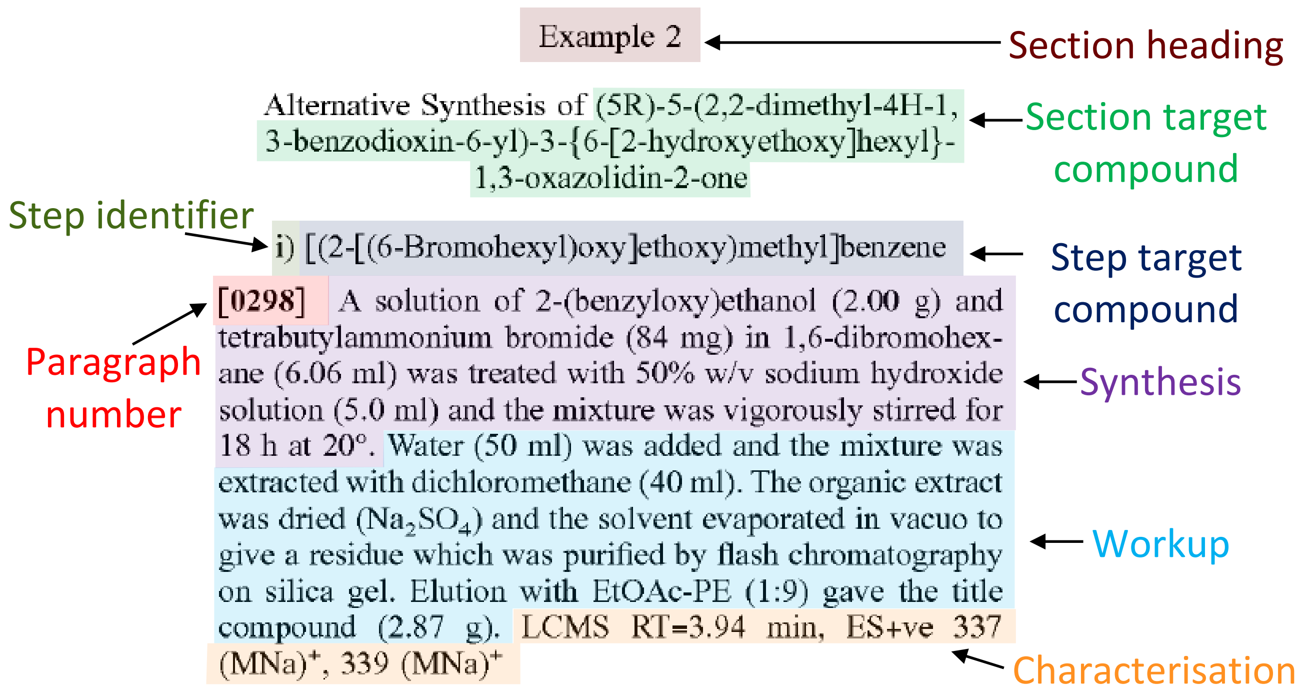
4.3 Corpus choice

USPTO patents were chosen for this task due to the ease of acquiring large numbers of them through Google Patents[131] and due to the absence of optical character recognition induced noise in post 1976 patents. Whilst USPTO patent applications are used throughout this chapter, the described system would be equally applicable to patent grant text and is also known to work with recent EPO patents due to the same XML tags being employed to designate headings and paragraphs. For evaluating the effect of changes and identifying areas of weakness in the reaction extraction system, a set of 106 patents that had been manually ascertained to contain reactions was formed from the USPTO patent applications for the first week of 2008. Patents from that week were not used when testing the final system.

4.4 Sectioning the relevant text within a patent

4.4.1 Archetypal experimental chemistry section

Experimental chemistry sections whilst still being free text are usually arranged in a predictable manner. Typically, they start with a heading indicating the compound to be synthesised, followed by a description of the synthesis, the workup steps undertaken and finally the characterisation of the compound. Where the synthesis of a compound necessitates the synthesis of intermediate compounds, typically each step of the synthesis is described separately with the final step giving the overall target compound. An example of the first step of an experimental section is shown in Figure 4-1, with its comprising sections annotated. A paragraph number is associated with each paragraph in USPTO and EPO patents and may be used to uniquely identify a paragraph within a given patent.



Example 2

Alternative Synthesis of (5R)-5-(2,2-dimethyl-4H-1,3-benzodioxin-6-yl)-3-{6-[2-hydroxyethoxy]hexyl}-1,3-oxazolidin-2-one

i) [(2-[(6-Bromohexyl)oxy]ethoxy)methyl]benzene

[0298] A solution of 2-(benzyloxy)ethanol (2.00 g) and tetrabutylammonium bromide (84 mg) in 1,6-dibromohexane (6.06 ml) was treated with 50% w/v sodium hydroxide solution (5.0 ml) and the mixture was vigorously stirred for 18 h at 20°. Water (50 ml) was added and the mixture was extracted with dichloromethane (40 ml). The organic extract was dried (Na_2SO_4) and the solvent evaporated in vacuo to give a residue which was purified by flash chromatography on silica gel. Elution with EtOAc-PE (1:9) gave the title compound (2.87 g). LCMS RT=3.94 min, ES+ve 337 $(MNa)^+$, 339 $(MNa)^+$

Figure 4-1 The start of a typical experimental section from a patent. The key features are annotated.

4.4.2 Sectioning workflow

Once a patent has been read in, the first challenge is to identify the experimental sections, which entails discriminating experimental chemistry text from non-experimental chemistry text. If a section is formed of multiple steps these must be associated with their parent section for the purpose of later allowing anaphora that reference particular sections/steps to be resolved. This process is shown schematically in Figure 4-2 .

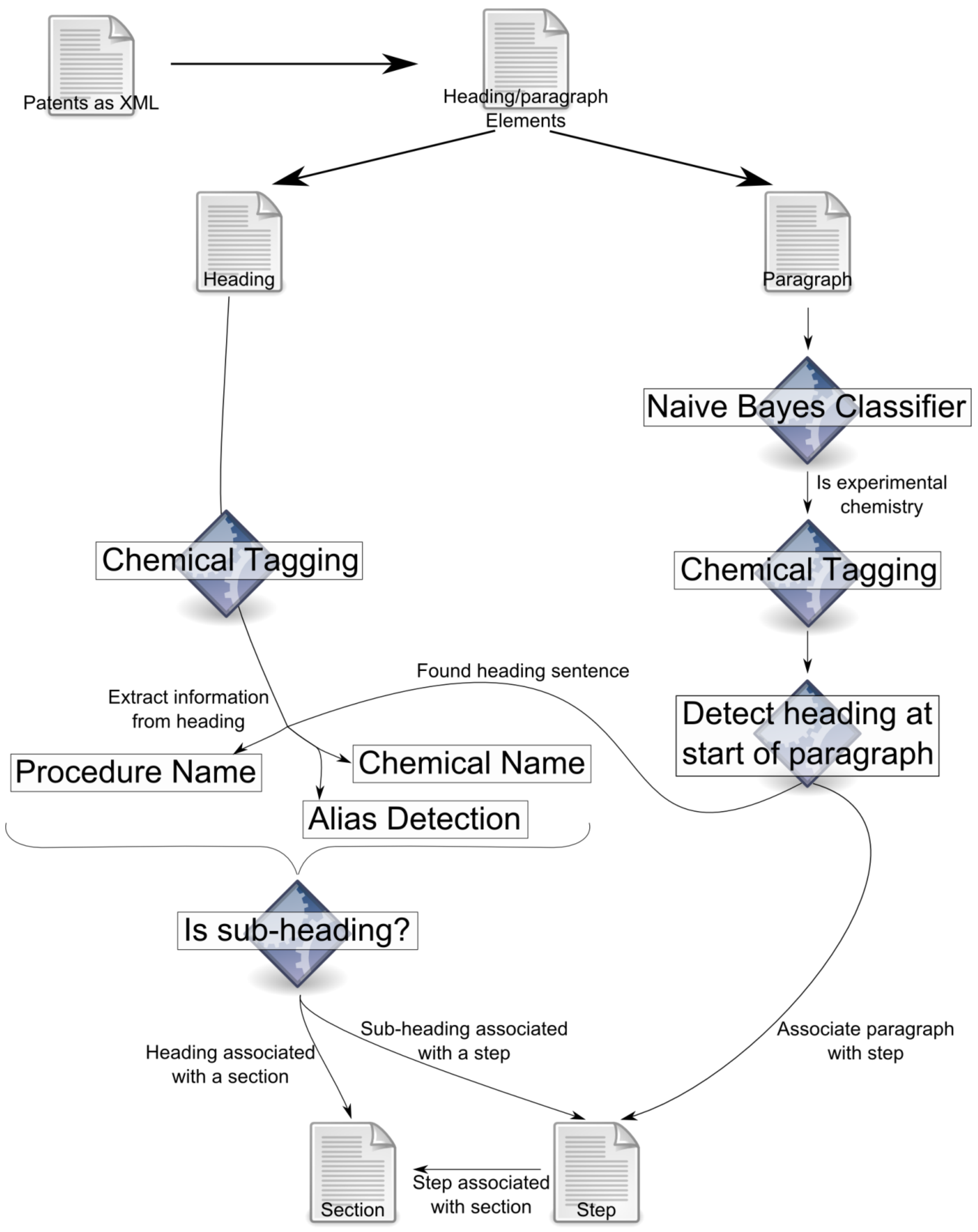

Patents as XML
Heading/paragraph Elements
Heading
Paragraph
Naive Bayes Classifier
Is experimental chemistry
Chemical Tagging
Chemical Tagging
Found heading sentence
Extract information from heading
Detect heading at start of paragraph
Procedure Name
Chemical Name
Alias Detection
Is sub-heading?
Sub-heading associated with a step
Heading associated with a section
Associate paragraph with step
Step associated with section
Section
Step

Figure 4-2 Schematic of processes employed in the segmentation of a document in steps, step headings and section headings

4.4.3 Identifying paragraphs and headings

The majority of headings and paragraphs are identifiable in the XML provided by the USPTO and EPO patent offices by the use of the element names `heading` and `p`. Headings that are present at the start of paragraphs are only detected after chemical tagging (cf. Section 4.4.6). Empirically it was found that paragraphs with ids starting with 'h-' followed by a number were often subheadings. Paragraphs matching this criterion were considered as sub-headings rather than as paragraphs when both a new line character was absent and ChemicalTagger found them to contain either a procedure name or a chemical name.

4.4.4 Paragraph classification

When a paragraph is encountered, determination of whether or not it is an experimental chemistry paragraph is made using a Naïve-Bayes classifier. This classifier is that previously described by Jessop et al.[168]. The classifier was trained by splitting a manually classified corpus of paragraphs evenly between training and testing. Once trained in this way, the classifier correctly identified 96.6% of experimental paragraphs as experimental and 89.9% of non-experimental as non-experimental in the test set.

For this work, the entire corpus of paragraphs was used to train the Bayesian classifier. Leave one out cross-validation gave results of 96.6% for identifying experimental and 90.7% for non-experiment paragraphs, indicating that the performance of this classifier is likely to be negligibly better than the one employed by Jessop.

4.4.5 Chemical tagging

The text of both headings and paragraphs are presented to ChemicalTagger to be marked up. The general operation of ChemicalTagger is described in Section 2.9. The output from ChemicalTagger is the primary input to the reaction extraction workflow and hence much effort has been made as part of this project to improve the output of ChemicalTagger. By improving ChemicalTagger it is also hoped that any other applications that rely on ChemicalTagger may benefit from the improvements that have been implemented.

4.4.5.1 Improved tokenisation

ChemicalTagger has a tokeniser interface which for experimental chemistry text is most well served by an implementation based on OSCAR4's tokeniser. Improvements were made to OSCAR4's tokeniser including the additions of more common abbreviations and correcting cases of chemical entities being erroneously split on hyphens and colons (Table 4-1).

Input	OSCAR 4.0.2	OSCAR 4.1
conc.	[conc][.]	[conc.]
2,2':6',2''-terpyridine	[2,2]['][:][6',2''-terpyridine]	[2,2':6',2''-terpyridine]
NH4OH(aq)	[NH4OH(aq)]	[NH4OH][(][aq][)]
D-glycero-D-manno-heptose	[D-glycero-D-manno][-][heptose]	[D-glycero-D-manno-heptose]

Table 4-1 Examples of improvements made to OSCAR's tokenisation

4.4.5.2 Improved robustness of sentence parser

When the ANTLR3 generated parser encounters input that is unacceptable to the grammar, whether due to being unlexable or not conformant to the grammar, input is skipped until an acceptable token may be consumed. The unrecognised input in such scenarios is, depending on the version of ChemicalTagger, either ignored completely or captured in an `UnmatchedPhrase`. The value of this element is the interleaved concatenation of the tokens and their tag values i.e. adjacent tokens may have been merged and the relationship between tags and tokens has been lost for the effected tokens. This undesirable behaviour can also lead to unexpected element content, if whilst in a rule no suitable input can be found e.g. a `MOLECULE` element without any elements corresponding to a chemical name.

To address this problem, the lexer was simplified to a whitespace tokeniser and the `Unmatched` alternative was expanded to cover all tags present in the grammar. The grammar is ordered such that this alternative is only tried, once all other rules for what a `Sentence` may contain have failed (Figure 4-3).

```
<Document>
  <Sentence>
    <Unmatched>
      <CC>but</CC>
    </Unmatched>
    <Unmatched>
      <NEG>not</NEG>
    </Unmatched>
    <NounPhrase>
      <MOLECULE>
        <OSCARCM>
          <OSCAR-CM>benzene</OSCAR-CM>
        </OSCARCM>
      </MOLECULE>
    </NounPhrase>
  </Sentence>
</Document>
```

Figure 4-3 Example of output from a phrase with tokens that may only be recognised by falling back to the `Unmatched` rule. The unexpected tokens are present in the output and remain associated with their tags.

4.4.5.3 Recognition of new concepts

Additions to ChemicalTagger's regex tagger and chemical sentence parser facilitated the recognition of yields, experimental procedures, chemical compound anaphora, pH conditions, the number of equivalents of a compound used and the physical state of a compound. To allow the detection of procedure/step names at the start of a paragraph that can be assumed to have that purpose only from context e.g. '1)', the grammar contains rules for detecting such cases that are applied specifically to the first phrase of input.

4.4.5.4 Improved recognition of existing concepts

Significantly more variants of units used to define quantities associated with reagents are now recognised. For example, improved tokenisation and recognition of the non-standard spelling 'mole' has corrected the exemplar issues given by Jessop[170]. The vocabulary for other terms recognised by ChemicalTagger e.g. yield verbs, has also been improved.

The recall and precision of reagents/products is affected most by the `MOLECULE` and `UNNAMEDMOLECULE` grammar rules. The former detects chemical entities with an associated name whilst the latter detects chemical entities that are defined purely by an anaphora. In both cases data, especially quantities such as amounts, volumes etc. must be contained within the rule so that the grammar will place them as children of the `MOLECULE`/`UNNAMEDMOLECULE` hence showing the association. Significant effort has been put into improving the coverage of these rules to attempt to mitigate problems with entities either not being recognised or not being associated with quantities that refer to them *cf*. Table 4-2.

Input	ChemicalTagger rev 166 (14/1/2011)	ChemicalTagger 1.3

sodium hydroxide solution (50ml)	`<NounPhrase>`
	`  <MOLECULE>`
	`    <OSCARCM>`
	`      <OSCAR-CM>sodium</OSCAR-CM>`
	`      <OSCAR-CM>hydroxide</OSCAR-CM>`
	`    </OSCARCM>`
	`  </MOLECULE>`
	`  <NN-CHEMENTITY>solution</NN-`
	`CHEMENTITY>`
	`  <QUANTITY>`
	`    <_-LRB->(</_-LRB->`
	`    <VOLUME>`
	`      <CD>50</CD>`
	`      <NN-VOL>ml</NN-VOL>`
	`    </VOLUME>`
	`    <_-RRB->)</_-RRB->`
	`  </QUANTITY>`
	`</NounPhrase>`
		`<NounPhrase>`
		`  <MOLECULE>`
		`    <OSCARCM>`
		`      <OSCAR-CM>sodium</OSCAR-CM>`
		`      <OSCAR-CM>hydroxide</OSCAR-CM>`
		`    </OSCARCM>`
		`    <NN-CHEMENTITY>solution</NN-`
		`CHEMENTITY>`
		`    <QUANTITY>`
		`      <_-LRB->(</_-LRB->`
		`      <VOLUME>`
		`        <CD>50</CD>`
		`        <NN-VOL>ml</NN-VOL>`
		`      </VOLUME>`
		`      <_-RRB->)</_-RRB->`
		`    </QUANTITY>`
		`  </MOLECULE>`
		`</NounPhrase>`

title compound as a colourless solid (52 mg, 23% yield)	`<NounPhrase>`
	`  <NN>title</NN>`
	`  <NN-CHEMENTITY>compound</NN-`
	`CHEMENTITY>`
	`</NounPhrase>`
	`<PrepPhrase>`
	`  <IN-AS>as</IN-AS>`
	`  <NounPhrase>`
	`    <DT>a</DT>`
	`    <JJ>colourless</JJ>`
	`    <NN-STATE>solid</NN-STATE>`
	`    <MIXTURE>`
	`      <_-LRB->(</_-LRB->`
	`      <MASS>`
	`        <CD>52</CD>`
	`        <NN-MASS>mg</NN-MASS>`
	`      </MASS>`
	`      <COMMA>,</COMMA>`
	`      <PERCENT>`
	`        <CD>23</CD>`
	`        <NN-PERCENT>%</NN-PERCENT>`
	`      </PERCENT>`
	`      <NN-YIELD>yield</NN-YIELD>`
	`      <_-RRB->)</_-RRB->`
	`    </MIXTURE>`
	`  </NounPhrase>`
	`</PrepPhrase>`
		`<UNNAMEDMOLECULE>`
		`  <JJ-COMPOUND>title</JJ-COMPOUND>`
		`  <NN-CHEMENTITY>compound</NN-`
		`CHEMENTITY>`
		`  <IN-AS>as</IN-AS>`
		`  <DT>a</DT>`
		`  <JJ>colourless</JJ>`
		`  <NN-STATE>solid</NN-STATE>`
		`  <QUANTITY>`
		`    <_-LRB->(</_-LRB->`
		`    <MASS>`
		`      <CD>52</CD>`
		`      <NN-MASS>mg</NN-MASS>`
		`    </MASS>`
		`    <COMMA>,</COMMA>`
		`    <YIELD>`
		`      <PERCENT>`
		`        <CD>23</CD>`
		`        <NN-PERCENT>%</NN-PERCENT>`
		`      </PERCENT>`
		`      <NN-YIELD>yield</NN-YIELD>`
		`    </YIELD>`
		`    <_-RRB->)</_-RRB->`
		`  </QUANTITY>`
		`</UNNAMEDMOLECULE>`

Table 4-2 Comparison of output from an older version of ChemicalTagger and the improved version

4.4.5.5 Improved action phrase assignment

The noun forms of certain verbs may in some cases be used to indicate an action e.g. ‘purification by gas chromatography’. Such phrases are now annotated with the action the noun confers; in this case the phrase is a “`Purify`” phrase.

4.4.5.6 Improved extensibility

In collaboration with the primary author of ChemicalTagger, Dr. Hawizy, changes were made to allow the program to accept an arbitrary number of taggers rather than having a hard coded expectation of an OSCAR4 tagger, regex tagger and a POS tagger. This is achieved by passing a list of taggers to ChemicalTagger, in which the position of the tagger in the list determines its priority.

For this work, the default regex tagger and POS tagger were used in conjunction with an OSCAR4 tagger customised with a small stop word list, an OPSIN tagger and a trivial chemical name tagger. The stop word list consisted of a small set of common mistakes that OSCAR4 was found to make on the evaluation set of patents.

The OPSIN tagger is an implementation of the OPSIN Document Extractor and is included primarily to identify cases where OSCAR4 might otherwise identify two entities within a single chemical name. A common cause of such problems is the presence of erroneous whitespace causing an apparently unmatched bracket to be tokenised separately from the rest of a chemical name. The OPSIN Document Extractor is presented with the untokenised input string and can often recognise chemical names containing erroneous whitespace as well as some complex chemical names incorrectly recognised by OSCAR4.

The trivial chemical name tagger is designed to recognise chemical names that OSCAR doesn't currently recognise and those for which the regex tagger produces competing tags e.g. 'Lawesson's reagent' in which reagent would be tagged as an NN-CHEMENTITY.

The prioritisation of the taggers is summarised in Table 4-3.

Priority	Tagger	Description
Highest	Trivial Chemical	Finds chemicals that neither OPSIN or OSCAR4 recognise
↓	OPSIN	Finds chemicals that are parsable by OPSIN
	Regex	Tags keywords e.g. yield words
	OSCAR4	Finds chemicals using a machine-learning approach
Lowest	OpenNLP	Tags part of speech

Table 4-3 Taggers employed and their priority

4.4.6 Identification of inline headings

Headings present at the start of paragraphs (Figure 4-4) must be detected and handled separately from the rest of the paragraph. This is achieved by examination of ChemicalTagger's output for the start of the paragraph. Constructs such as a procedure identifier followed by a suitable delimiter and phrases following patterns like “Synthesis of xxx” are identified as headings and removed from the paragraph. All patterns operate on ChemicalTagger’s tags to allow more lexical variations to be accepted. For example the “Synthesis of xxx” pattern would be implemented as an examination of an initial `NounPhrase` for an `NN-SYNTHESIZE` tag followed by a `PrepPhrase` element containing an `IN-OF` tag and a `NounPhrase` element. Identified inline headings are then treated analogously to other headings.

4-Fluoro-2-methylbenzonitrile (31). A mixture of 2-bromo-5fluorotoluene (3.5 g, 18.5 mmol) and…

Figure 4-4 Example of a paragraph containing an inline heading (bold text)

4.4.7 Processing of headings

After a heading has been run through ChemicalTagger it is examined for molecule entities and procedure names. Entities known to present as false positives in OSCAR4’s output are filtered out using the regexes used for identifying molecules as being of type “false positive” (*cf*. Section 4.5.1.4). Additionally, as OSCAR4 is known to classify strings of capital letters as chemicals, if the entirety of the heading is formed of capital letters e.g. ‘ABSTRACT’, no molecule entities are recognised. If the heading has a molecule entity and/or a procedure name the heading is assumed to be part of an experimental section; otherwise the heading serves as a delimiter between experimental sections.

A procedure name is either associated with an experimental section or a reaction step dependent on whether it is believed to be a sub-heading. A procedure name is determined to be a sub-heading if it contains neither an `NN_METHOD` nor `NN_EXAMPLE` word or the procedure’s NN_METHOD word is ‘stage’ or ‘step’ (Table 4-4). As previously mentioned, paragraphs with ids starting with ‘h-’ are treated as sub-headings.

Example of heading procedure names	Examples of sub-heading procedure names
Example 5	Step b
General procedure 3	1)
Method 2a	2.

Table 4-4 Examples of headings and subheadings

If a molecule entity is detected in a heading an attempt is made to identify a string within the heading that appears to be an alias for the compound so that subsequent use of the alias may resolve to that compound. A heading molecule is associated with a reaction step or an experimental section dependent on whether or not an appropriate procedure name had been found indicating the start of a reaction step.

4.4.8 Processing of paragraphs

Paragraphs, which were classified as experimental, are associated with the current reaction step, when the current step or current experimental section, is associated with either a molecule entity or a procedure name. The requirement of an appropriate preceding heading allows further non-experimental paragraphs that passed though the paragraph classifier to be ignored. As a special case, paragraphs containing a yield phrase within which resides a molecule entity are always added to the current reaction step to allow for the case where the paragraph fully describes a reaction whilst not being preceded by a heading.

4.5 Section Parsing

The identified sections are processed sequentially in the order that they were defined in the patent using the scheme in Figure 4-5.

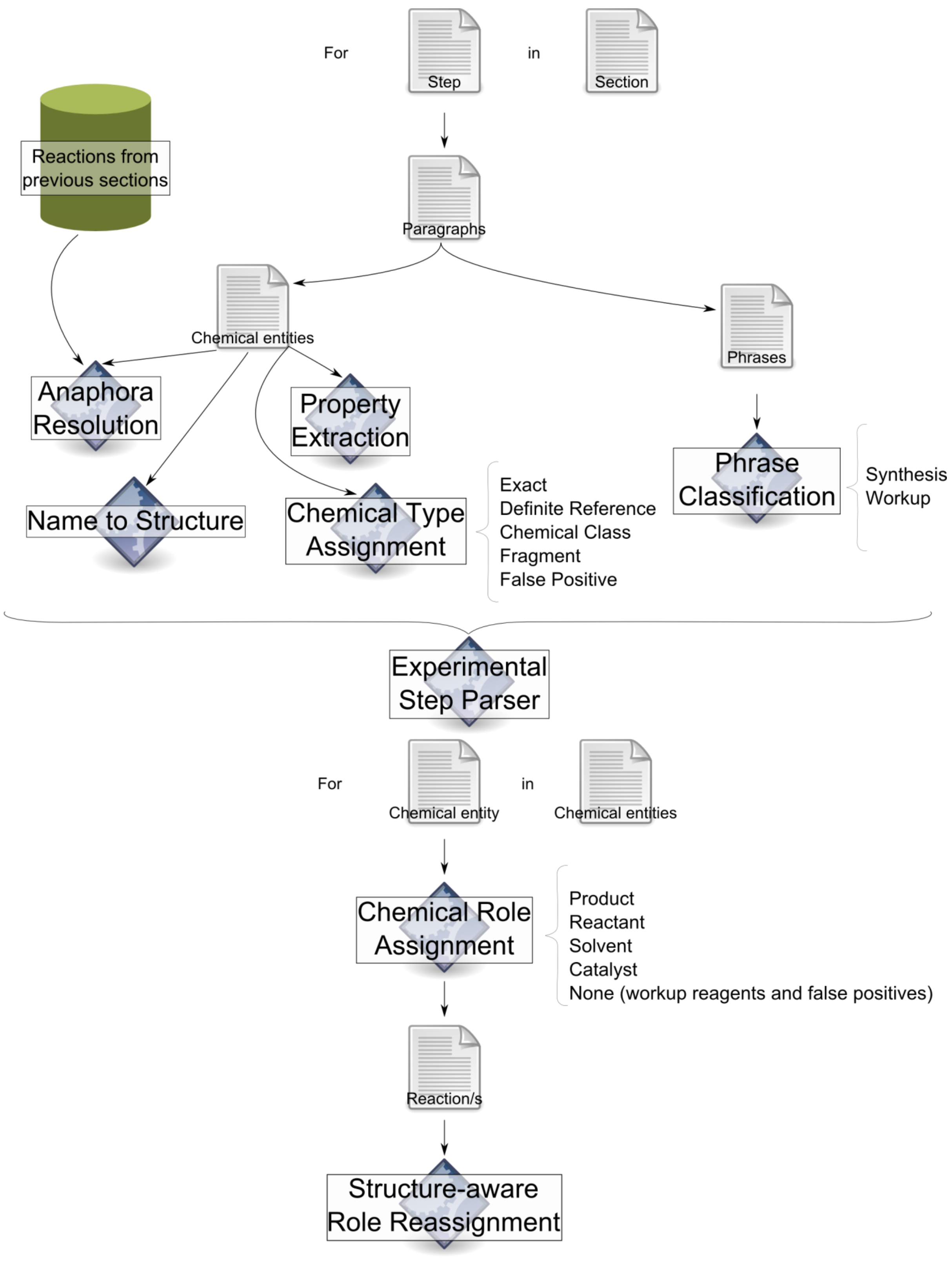


Figure 4-5 Schematic of processes employed to extract from reactions from an experimental chemistry section.

4.5.1 Processing of chemical entities

4.5.1.1 Name to structure

The workflow relies on OSCAR4.1 for the resolution of chemical names, which in turn relies on OPSIN 1.2.0, the chemical names present in the ChEBI database as of December 2011 and a manually created dictionary of chemical formulae and common chemical abbreviations. This latter dictionary was increased from 81 entries to 260 entries to afford better coverage of the abbreviations used for common reagents in organic chemistry.

To allow better support for the combination of a systematic name with an adjacent abbreviated name, where such cases are identified by the presence of a non-chemical hyphen, the different parts of the name are handled separately and the SMILES and InChI then constructed by merging the output for the two names. Merging of InChIs is achieved using the InChI library by constructing an input containing all the structures. In the case where a name is uninterpretable and has no delimiters identified by ChemicalTagger, if the name is found to contain exactly one slash or dot or space, parsing of the substrings either side of the delimiter is attempted.

4.5.1.2 Anaphora identification and resolution

Chemical entities may be referred to by anaphora; that is terms that reference a previous entity. Four types of anaphora are recognised: references to compound identifiers, references to procedures, textual aliases and textual references to heading compounds.

A reference to a compound identifier is identified in ChemicalTagger's output by the encapsulating `REFERENCETOCOMPOUND` element (Figure 4-6).

```
<UNNAMEDMOLECULE>
  <NN-CHEMENTITY>compound</NN-CHEMENTITY>
  <REFERENCETOCOMPOUND>
    <CD>92</CD>
  </REFERENCETOCOMPOUND>
  <QUANTITY>
    <_-LRB->(</_-LRB->
    <MASS>
      <CD>107</CD>
      <NN-MASS>mg</NN-MASS>
    </MASS>
    <COMMA>,</COMMA>
    <AMOUNT>
      <CD>0.24</CD>
      <NN-AMOUNT>mmol</NN-AMOUNT>
    </AMOUNT>
    <_-RRB->)</_-RRB->
  </QUANTITY>
</UNNAMEDMOLECULE>
```

Figure 4-6 Compound 92 in this example is a reference to a previously defined chemical entity

A reference to a procedure is identified by the encapsulating `PROCEDURE` element (Figure 4-7). Only procedures mentioned within a molecule are assumed to be the source of the chemical entity.

```
<MOLECULE>
  <OSCARCM>
    <OSCAR-CM>Chloropyrimidine</OSCAR-CM>
  </OSCARCM>
  <QUANTITY>
    <_-LRB->(</_-LRB->
    <MASS>
      <CD>0.5</CD>
      <NN-MASS>g</NN-MASS>
    </MASS>
    <COMMA>,</COMMA>
    <AMOUNT>
      <CD>1.75</CD>
      <NN-AMOUNT>mmol</NN-AMOUNT>
    </AMOUNT>
    <_-RRB->)</_-RRB->
  </QUANTITY>
  <IN-FROM>from</IN-FROM>
  <PROCEDURE>
    <NN-METHOD>step</NN-METHOD>
    <_-LRB->(</_-LRB->
    <NN-IDENTIFIER>c</NN-IDENTIFIER>
    <_-RRB->)</_-RRB->
  </PROCEDURE>
</MOLECULE>
```

Figure 4-7 Chloropyrimidine in this example is an anaphora for a particular chloropyrimidine from a previous step.

If a chemical entity is associated with a bracketed chemical entity the two are assumed to be synonyms (Figure 4-8). As the purpose of this is to improve recall if the synonym is subsequently used, only cases in which one chemical entity is resolvable to a structure but the other is not are considered. Subsequent mentions of the unresolvable name will yield the same structure as the resolvable name.

```
<MOLECULE>
  <OSCARCM>
    <OSCAR-CM>N-Ethoxycarbonyl-2-ethoxy-1,2-dihydroquinoline</OSCAR-CM>
  </OSCARCM>
  <OSCARCM>
    <_-LRB->(</_-LRB->
    <OSCAR-CM>EEDQ</OSCAR-CM>
    <_-RRB->)</_-RRB->
  </OSCARCM>
</MOLECULE>
```

Figure 4-8 Example of a systematic chemical name and its abbreviation

Entities with a name matching the case insensitive regex:

(crude|desired|title[d]?|final|aimed|expected|anticipated) (compound|product) are assumed to refer to heading compounds. Typically, this is the compound associated with the heading of the current step. However, if this is the final step, or the step is not associated with a compound, then the current section heading compound is assumed.

4.5.1.3 Property Extraction

Where present volumes, amount (i.e. number of mols), mass, molarity (i.e. concentration), number of equivalents, pH, percent yield and the physical state of a compound may be extracted from ChemicalTagger's output. Association of these properties with a chemical entity is achieved by the relevant elements being nested within the chemical entity in the ChemicalTagger output.

4.5.1.4 Chemical type assignment

Every chemical entity is assigned a type (Table 4-5).

Chemical Entity Type	Description	Examples
exact	Describes a specific compound	2-chloroethanol, pyridine
definite reference	Describes a specific compound but relies on information described elsewhere in the document	Compound 5, the pyridine from example 2
chemical class	Describes a series of compounds	ether, pyridines
fragment	Describes a radical or substructure of a compound	ethyl, pyridine ring
false positive	Not a chemical entity or one that would not be expected to be part of a chemical reaction (e.g. an NMR solvent)	CDCl3, TLC

Table 4-5 Description of chemical entity types assigned by the system

False positives are recognised by the entities presence within an `APPARATUS` or `AtmospherePhrase` phrase (as identified by chemical tagging) or being followed by a word indicating the chemical entity is a surface e.g. 'silica surface'. Additionally a series of regular expressions are used to match NMR solvents as well as characterisation terms known to be misidentified by OSCAR4 as chemicals.

Entities are recognised as being of type "chemical class" by being prefixed by the determiners 'a' or 'an', by being followed by the word 'compound' or 'derivative', by being a known function class e.g. 'aldehyde', by being assigned as such by the OPSIN Document Extractor or by ending in a plural ending.

Entities are recognised as type “fragment” if they are followed by words like ‘group’ or ‘ring’ or are assigned as such by the OPSIN Document Extractor e.g. ‘ethyl’.

Any entity not explicitly assigned a type is assumed to be of type “exact”.

4.5.2 Identification of discourse type

Paragraphs are broken down into phrases by the top level phrase elements into which ChemicalTagger has grouped the input. Each phrase is then classified as either synthesis or workup. Phrases that form the characterisation section are not explicitly identified as the boundary between workup and characterisation may occur within a phrase complicating exact identification of the boundary. As it is practical to filter out the vast majority of chemical entities that are associated with characterisation, characterisation sections are indirectly ignored by virtue of contributing no allowed chemical entities.

The approach used to identify the discourse type assumes that all text up to the start of the workup section relates to synthesis and hence discourse analysis concentrates on the identification of phrases that relate to workup. It was found that phrases of types "`Concentrate`", "`Degass`", "`Dry`", "`Extract`", "`Filter`", "`Partition`", "`Precipitate`", "`Purify`", "`Recover`", "`Remove`", "`Wash`", "`Quench`" were associated with workup. Where a phrase does not fit into one of these roles, the assumption is made that the phrase is of the same type (synthesis/workup) as the preceding phrase.

In contrast to the literature solutions, the presence of a molecule possessing an associated amount, yield or number of equivalents is used to indicate the return to a synthesis section. This heuristic arises from the observation that the amounts of workup reagents are rarely precisely specified and allows the support for multi-step reactions within the same paragraph. Additionally the presence of a "`Synthesize`" or "`Yield`" phrase is assumed to indicate the end of a workup section.

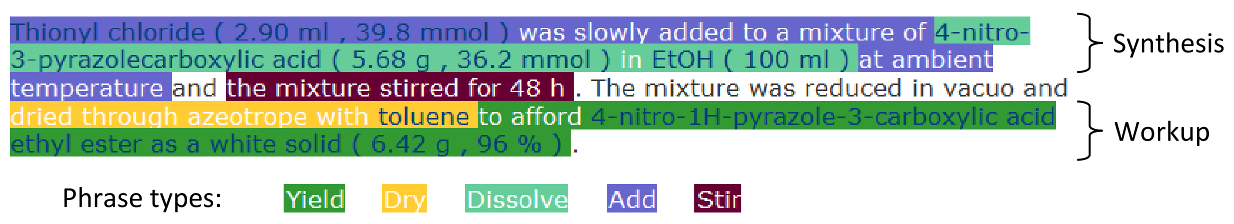

Thionyl chloride (2.90 ml , 39.8 mmol) was slowly added to a mixture of 4-nitro-3-pyrazolecarboxylic acid (5.68 g , 36.2 mmol) in EtOH (100 ml) at ambient temperature and the mixture stirred for 48 h . The mixture was reduced in vacuo and dried through azeotrope with toluene to afford 4-nitro-1H-pyrazole-3-carboxylic acid ethyl ester as a white solid (6.42 g , 96 %) .

Figure 4-9 Typical experimental chemistry paragraph showing the different phrase types identified by ChemicalTagger. In this case the dry phrase is used to indicate the start of the workup section.

4.5.3 Chemical role assignment

A putative role in the reaction is assigned for each chemical entity that has not been excluded due to being in a workup section or being of type “false positive” (Table 4-6).

Chemical role	Description
product	This is a compound produced as a result of a reaction
reactant	A substance that undergoes a chemical change in a reaction
solvent	A compound in which reactants are dissolved
catalyst	A compound which is not consumed by a reaction and accelerates a reaction

Table 4-6 Roles considered for chemical entities in a reaction

Role assignment is achieved through a mixture of the output of ChemicalTagger, analysis of the local textual environment and lists of known solvents/catalysts.

4.5.3.1 Product Role

A chemical entity is assigned as being a product in these situations:

- It is associated with a percentage yield
- It is part of a noun phrase followed by ‘is synthesised’ (or a similar phrase)
- It is part of a `yield` phrase
- It is identified as being an anaphora to the current heading compound e.g. ‘title compound’

4.5.3.2 Reactant Role

This is the default role assigned if no clear indication of an alternate role can be determined from the text or textual environment.

4.5.3.3 Solvent Role

A chemical entity is assigned as being a solvent in these situations:

- ChemicalTagger assigns it as a solvent
- It corresponds to an InChI-less solvent e.g. brine
- It is proceeded by the words ‘in’, ‘in a mixture of’ or either of these followed by a chemical entity and the word ‘and’

Once a reaction has been constructed, sensibility checks also ensure that if a reagent is listed as both a solvent and reactant that all instances are reclassified as a solvent. Additionally if a reaction does not have a solvent, a reagent without a specified amount may be assigned as a solvent if its InChI matches that of a known solvent or its volume is imprecisely defined.

4.5.3.4 Catalyst Role

A chemical entity is assigned as a catalyst in these situations:

- ChemicalTagger assigns it as a catalyst
- Its name corresponds to a known catalyst
- Its InChI corresponds to a known catalyst

A chemical entity that is found to contain a transition metal atom which is absent from the product molecule/s is considered to be a catalyst with a few exceptions for where a transition metal is part of an oxidising agent and organocopper/zinc/mercury chemistry. These two exceptions are enforced by identifying particular transition metals in high oxidation states and the presence of carbon-metal bonds respectively.

4.6 Reaction mapping

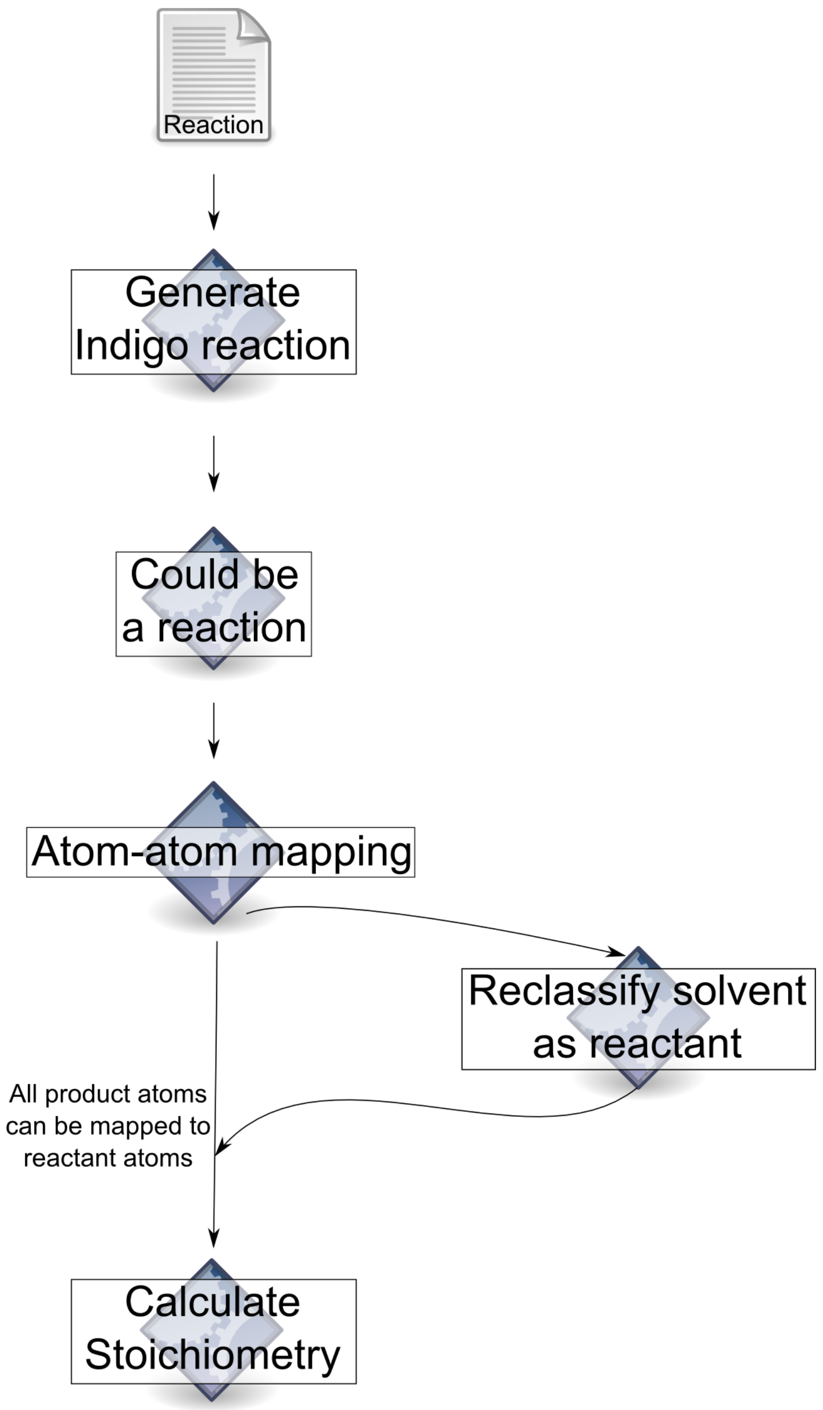


Figure 4-10 Schematic of processes employed in converting putative reactions to atom-mapped reactions

4.6.1 Indigo reaction creation

The extracted reactions are loaded into the Indigo toolkit (version 1.1-beta9) to provide atom-mapping and depiction. This is accomplished using SMILES as the input format. To allow more efficient atom-mapping and more aesthetic depictions, for each role, chemical entities that share the same InChI are considered to be the same compound and hence are only added once to the Indigo reaction.

Upon creation of the Indigo reaction, a check is done to ensure that the reaction has a product, a total of at least two reactants/solvents/catalysts, and that none of the reactants have the same structure as the product. It should be noted that these conditions may fail for correctly identified reactions if SMILES could not be obtained for reactants and/or product.

4.6.2 Atom-atom mapping

In a well formed chemical reaction all atoms in the product/s must have come from the reactants and hence any “reactions” for which this is not true should be rejected. One way of achieving this is by performing atom-atom mapping (AAM). This is a technique for relating the atoms of the reactants to those of the product. This is typically implemented using a maximum common subgraph algorithm to find the maximum number of atoms in the product that may be accounted for by a given reactant. The resultant mapping is not necessarily unique in terms of the atoms picked within a reactant or even in terms of which reactants are used to provide atoms.

By default, Indigo attempts to match atoms with identical charge and valency in the reactant and products and similarly attempts to match bonds with the same bond order. Hence for greater leniency and to reflect some of the operations that may occur in real chemical reactions these conditions are relaxed to allow changes in charge, valency and bond order.

In some reactions it was found that the solvent was also a reactant. To prevent such cases resulting in incomplete atom mapping when atom mapping fails, an attempt is made to reclassify a solvent as a reactant and AAM is repeated. Currently there is no method for reporting this dual role and instead the solvent will be reported as a reactant.

4.6.3 Stoichiometry calculation

Where AAM was successful it may be used to calculate the stoichiometry of the reaction. The stoichiometry of each reactant is assumed to be equal to the greatest number of times a particular atom from the reactant appears in the product. This approach has several limitations; namely, that a reactant will not be considered to contribute to the reaction if it either only contributes non-heavy atoms or only contributes to an unstated side product and that the atom mappings may be wrong, especially when the system has identified too many reactants.

4.6.4 Output

The list of all reactions and a list of mappable reactions are retrievable after a patent has been processed. These reactions may be serialised to a graphical depiction (Figure 4-11) and CML (Figure 4-12). For mappable reactions, the graphical depiction will contain the results of the AAM.

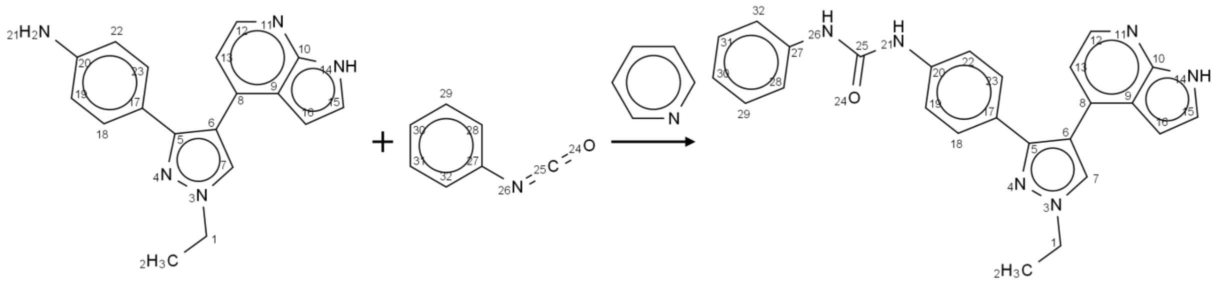


Figure 4-11 Graphical depiction of an extracted reaction. The numbers indicate the mapping between atoms in the reactants and products. The solvent is present above the arrow.

```
<?xml version="1.0" encoding="UTF-8"?>
<reaction 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/"
xmlns:unit="http://www.xml-cml.org/unit/" xmlns:cml="http://www.xml-cml.org/schema"
xmlns:dl="http://bitbucket.org/dan2097">
  <dl:reactionSmiles>[CH2:1]([n:3]1[cH:7][c:6](-
[c:8]2[cH:13][cH:12][n:11][c:10]3[nH:14][cH:15][cH:16][c:9]23)[c:5](-
[c:17]2[cH:23][cH:22][c:20]([NH2:21])[cH:19][cH:18]2)[n:4]1)[CH3:2].[O:24]=[C:25]=[N:26][c:27]
1[cH:32][cH:31][cH:30][cH:29][cH:28]1&gt;c1cc[n]cc1&gt;[CH2:1]([n:3]1[cH:7][c:6](-
[c:8]2[cH:13][cH:12][n:11][c:10]3[nH:14][cH:15][cH:16][c:9]23)[c:5](-
[c:17]2[cH:23][cH:22][c:20]([NH:21][C:25]([NH:26][c:27]3[cH:32][cH:31][cH:30][cH:29][cH:28]3)=
[O:24])[cH:19][cH:18]2)[n:4]1)[CH3:2]</dl:reactionSmiles>
  <productList>
    <product role="product">
      <molecule id="m0">
        <name dictRef="nameDict:unknown">title product</name>
      </molecule>
      <amount units="unit:percent yield">50.0</amount>
      <identifier dictRef="cml:smiles"
value="C(C)N1N=C(C(=C1)C1=C2C(=NC=C1)NC=C2)C2=CC=C(C=C2)NC(=O)NC2=CC=CC=C2"/>
      <identifier dictRef="cml:inchi" value="InChI=1/C25H22N6O/c1-2-31-16-22(20-12-14-26-24-
21(20)13-15-27-24)23(30-31)17-8-10-19(11-9-17)29-25(32)28-18-6-4-3-5-7-18/h3-
16H,2H2,1H3,(H,26,27)(H2,28,29,32)"/>
      <dl:entityType>definiteReference</dl:entityType>
      <dl:state>powder</dl:state>
    </product>
  </productList>
  <reactantList>
    <reactant role="reactant" count="1">
      <molecule id="m1">
        <name dictRef="nameDict:unknown">4-[1-ethyl-4-(1H-pyrrolo[2,3-b]pyridin-4-yl)-1H-
pyrazol-3-yl]aniline</name>
      </molecule>
      <amount units="unit:mmol">2.2</amount>
      <identifier dictRef="cml:smiles"
value="C(C)N1N=C(C(=C1)C1=C2C(=NC=C1)NC=C2)C2=CC=C(N)C=C2"/>
      <identifier dictRef="cml:inchi" value="InChI=1/C18H17N5/c1-2-23-11-16(14-7-9-20-18-
15(14)8-10-21-18)17(22-23)12-3-5-13(19)6-4-12/h3-11H,2,19H2,1H3,(H,20,21)"/>
      <dl:entityType>exact</dl:entityType>
    </reactant>
    <reactant role="reactant" count="1">
      <molecule id="m2">
        <name dictRef="nameDict:unknown">phenyl isocyanate</name>
      </molecule>
      <amount units="unit:mmol">2.4</amount>
      <identifier dictRef="cml:smiles" value="O=C=Nc1ccccc1"/>
      <identifier dictRef="cml:inchi" value="InChI=1/C7H5NO/c9-6-8-7-4-2-1-3-5-7/h1-5H"/>
      <dl:entityType>exact</dl:entityType>
    </reactant>
  </reactantList>
  <spectatorList>
    <spectator role="solvent">
      <molecule id="m3">
        <name dictRef="nameDict:unknown">pyridine</name>
      </molecule>
      <amount units="unit:mL">4</amount>
      <identifier dictRef="cml:smiles" value="c1ccncc1"/>
      <identifier dictRef="cml:inchi" value="InChI=1/C5H5N/c1-2-4-6-5-3-1/h1-5H"/>
      <dl:entityType>exact</dl:entityType>
    </spectator>
  </spectatorList>
</reaction>
```

Figure 4-12 CML output for the extracted reaction depicted in Figure 4-11

The CML output includes all the information extracted for a given reaction. For each chemical entity this includes a role, an entity type (*cf*. Section 4.5.1.4), and where possible a chemical structure (as SMILES and InChI), quantities e.g. volumes, amounts, weights etc. and the physical state. If available the yield of the product is also recorded. Where AAM was successful the atom-mapped reaction is included as reaction SMILES and the stoichiometry of the reactants are captured by the `count` attribute with reactants that contribute no atoms to the products lacking a `count` attribute.

4.7 Evaluation

4.7.1 Methodology

USPTO patent applications for the period of 2008 through to the end of 2011 were downloaded from Google Patents[131]. The XML representation of the patents was inspected to determine the IPC (International Patent Classification) codes associated with each patents. Only patents containing the IPC code 'C07' were selected for processing. The 'C' refers to section C which describes chemistry and metallurgy whilst the '07' refers to sub category of organic chemistry. A patent is associated with one or more IPC codes. Additionally patents from the first week of 2008 were not used as these were used to identify limitations in older versions of the reaction extraction system.

This yielded a set of 65,034 patents on which the reaction extraction system was ran. For each patent the reactions were serialised to depictions and CML with segregation of the output based on whether or not AAM was successful. The file names of the serialised reactions include the paragraph from which the reactions were extracted.

Whilst a successful atom mapping is a good indicator that a found reaction really is a reaction other aspects of the output can be used to filter out dubious reactions hence additional criteria were applied to produce a smaller but higher quality set of reactions. These were:

- Reactions containing any products that could not be resolved to structures were excluded. This helps with some cases where the product is not resolved to a structure but instead the counter ion from a salt is resolved. Cases where a product is described in such a way that ChemicalTagger associates both a `MOLECULE` and an `UNNAMEDMOLECULE` with different parts of the product's description may be unnecessarily excluded by this criterion.
- Reactions containing any entities of types: fragment or chemical class, were excluded in order to exclude generic rather than specific reactions.

A sample of 100 randomly selected reactions was selected from this set to evaluate the quality of the extracted reactions. For each selected reaction, chemical entities were manually identified and associated with a role. If this role was not that the entity was a workup/characterisation reagent then the entity type and quantities, that the reaction extraction system attempts to find, were also manually identified.

The correctness of name to structure conversion was not evaluated as it is likely to be more accurate than manual conversion by the average chemist. Cases where the reaction extraction system missed reagents that were only implicitly described, for example, in a reaction being performed analogously to a previous reaction, were not penalised as analogous reactions are outside of the scope of the system as implemented.

4.7.2 Results

4.7.2.1 Errors encountered

Using v1.0 of the reaction extraction system, 10 of the 65,034 patents had to be manually skipped due to either crashing the reaction extraction system (3 cases) or taking an unacceptably long time to complete (7 cases). The results presented in this chapter are hence for the other 65,024 patents.

One crash was caused by an oversight in the way OPSIN generates parse combinations. This occurred when processing a long series of fragments that had been erroneously identified as a single name and resulted in an OutOfMemoryError. Another was caused by a StackOverflowError when ChemicalTagger attempted to parse an exceptionally long sentence of bracketed molecules which would each be associated with the previous in the parse tree. The other crash was caused by an OutOfMemoryError when tagging a nearly 700,000 character long “sentence”. All the cases in which a patent took an unacceptably long time to complete related to the AAM procedure. A timeout of 1 minute was specified for the AAM but a bug in Indigo-1.1-beta9 meant that a small minority of reactions did not respect the timeout. This was reported to the developers of Indigo and fixed in Indigo-1.1.

Using the subsequently released version of Indigo, fixing the bug in OPSIN and the OPSIN Document Extractor, and limiting paragraphs/headings to 35,000 characters allowed the system to run over all patents in the four year period without any manual intervention. The process took 84 hours using 1 thread for each year on an Intel Core i7-2600k. This is sufficiently fast to easily allow the patent applications for a week to be analysed within a day of their public release.

4.7.2.2 Overall statistics

484,259 atom mapped reactions were extracted (Figure 4-13), of which 424,621 met the more stringent criteria described in 4.7.1.

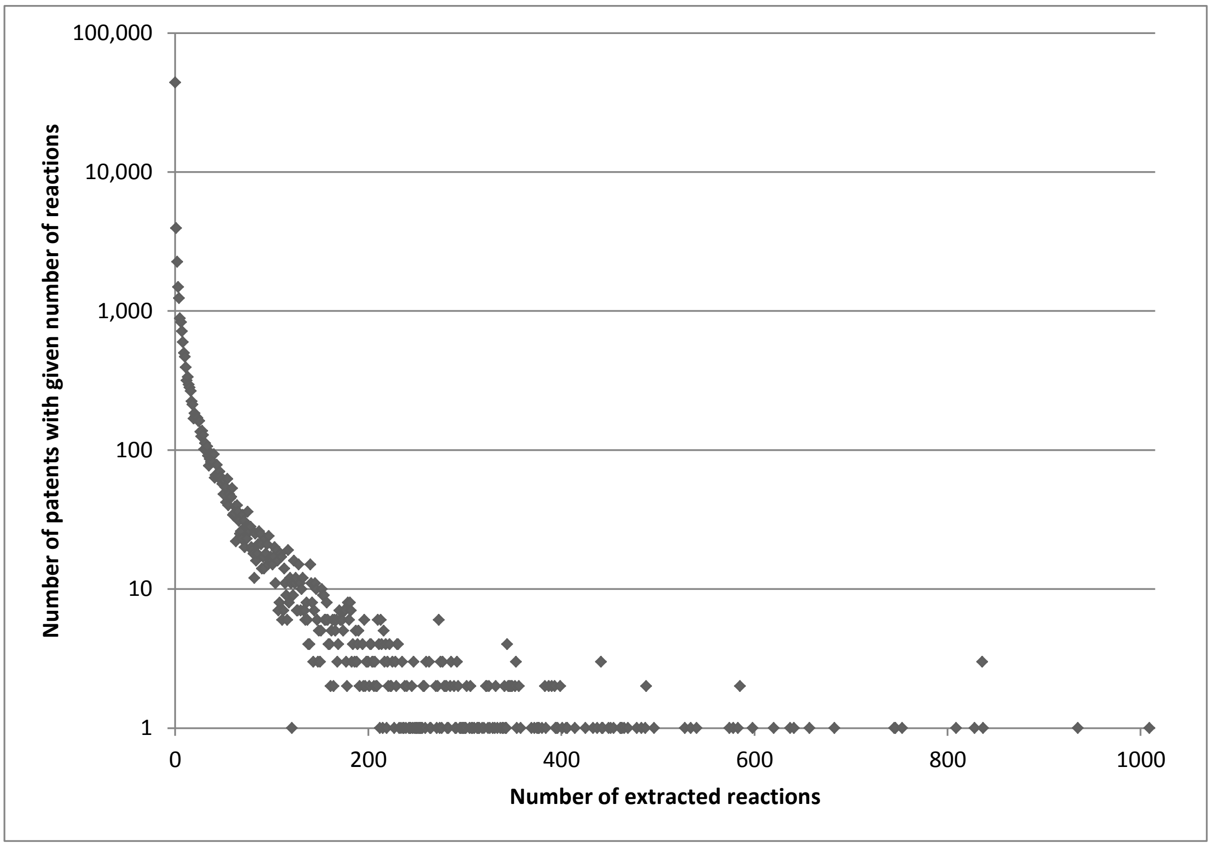


Figure 4-13 Number of patents with a given number of atom mapped reactions

4.7.2.3 Evaluated reaction quality

Table 4-7 indicates the precision/recall with which the entities involved in a chemical reaction were identified. False positives may be workup reagents, characterisation reagents or not chemicals at all. False negatives are those entities which are involved in the reaction but were not identified.

True Positives	474
False Positives	60
False Negatives	18
Precision	88.9%
Recall	96.4%
F_1 score	92.5%

Table 4-7 Statistics for recognition of chemical entities (reagents and products)

Table 4-8 shows whether for correctly detected chemical entities with quantities specified in the text, whether these were associated with the chemical entity. A quantity in this context could be a yield, amount, weight, volume etc. If an entity has multiple quantities all must be correctly associated to be considered a success.

Agent type	Successful cases/total cases (%)
Reagents	317/321 (98.8%)
Products	48/74 (64.9%)

Table 4-8 Statistics for association of quantities with reagents/products possessing quantities

Table 4-9 shows for each chemical entity of a given role in the manually annotated reactions (which is also found in the automatically extracted reactions) whether the extracted entity has that role.

Role	Successful cases/total cases (%)
Product	99/100 (99.0%)
Reactant	241/244 (98.8%)
Solvent	85/99 (85.9%)
Catalyst	10/24 (41.7%)
Other Spectator	0/7 (0%)
Overall	435/474 (91.8%)

Table 4-9 Statistics for association of roles with entities

4.8 Discussion

The number of patents containing a specified number of extracted reactions (Figure 4-13) shows a power law distribution with respect to the number of reactions extracted from each patent. The majority of patents have less than 10 reactions but a significant minority have greater than 100. Patents with greater than 100 reactions account for only 1.59% of the patents processed but held 41.26% of the extracted reactions.

Recall of chemical entities (Table 4-7) was high (96.4%), whilst precision was somewhat lower. This was primarily due to the classification of workup reagents, especially the first workup reagent, as reactants. This is due to the text often just saying that the reagent was added without any indication of the purpose. Heuristics involving the addition of a common solvent as the last step of a reaction could be investigated.

Association of quantities (Table 4-8) with reagents was near perfect and can be considered a solved problem. There appear to be only a finite number of ways in regular use for associating quantities with chemical entities and all of them are supported by ChemicalTagger. Association of quantities with the product was less successful. This was because this information is often present at the end of an experimental section and only implicitly assumed to apply to the product. Heuristics could be investigated for improving association of unassigned quantities with the product of the reaction. This is expected to be especially applicable to unassigned yields as these will almost invariably be the yield of the reaction.

Assignment of roles (Table 4-9) was excellent for products (99.0%) and reactants (98.8%) but worse for spectator reagents. For this evaluation, the definition of a catalyst was the strict definition that the reagent is not consumed in the course of the reaction. As experimental descriptions typically only describe the intended product and not the fate of the other reagents involved it is often impossible to tell from just the text whether or not a reagent is a catalyst. This also made the assignment of reagents as catalysts problematic for the manual annotations as the likely mechanism of the reaction had to be investigated in some cases. Similarly, a reagent was considered a reactant even if it did not contribute any heavy atoms to the product as long as it was believed to be consumed by the reaction. Nonetheless, despite the difficulties in identifying catalysts the addition of more known catalysts and the application of heuristics based around the relative quantity of reagent used would yield improved results.

Overall only 22% of the extracted reactions were flawless by all the metrics evaluated i.e. perfect entity recognition, quantity assignment and role assignment. However it should be borne in mind that most failures were minor, for example: a yield not assigned to a product, a workup reagent assigned as a reactant, etc. It should also be noted that some mistakes in chemical entity identification are not visible in the graphic depiction. This happens when two copies of a reagent are inadvertently identified (as happens if both a reagent and an anaphora to the reagent are independently resolved), as they will have been merged (using InChI to check for identity) prior to depiction.

The correct identification of product and major starting material is a somewhat more qualitative but potentially more useful metric, especially for reaction searching, with the proviso that there are no false positive entities that could be mistaken for either of these entities. This was true of 95% of the evaluated reactions. There were several reasons for the failure with the other 5%. A recurring problem was the difficulty in determining the meaning of a reference to a procedure as it could mean:

- The compound produced at the end of that procedure
- The compound produced at the end of that procedure but only in the context of an analogy to the current reaction

- The procedure itself

One failure involved a reaction being erroneously split into two reactions due to ChemicalTagger misassigning 'starting' as a verb rather than as an adjective (in the context of 'to give (i) starting material'). This caused the yield phrase to end at the word 'starting' and hence the true products ended up as the reactants for a second reaction. Another failure involved a reaction in which some of the compounds were defined using anaphora to previously defined labelled compounds. In this case the association between these previous compounds and their numeric identifiers had not been made; hence preventing resolution of these anaphoric references.

4.9 Comparison to other approaches

PatentEye (Section 4.2.2) was evaluated, by Jessop, on ten weeks of EPO patents which corresponded to a corpus of 667 patents. From these 4444 reactions were extracted of which a subset was evaluated to assess the recall and precision of reagent, and recall of product identification. The results were 64% recall and 78% precision for reagent identification with the criteria for a true positive being that both the entity and associated quantities were found. Table 4-7 in conjunction with the 98.8% association of quantities with reagents suggests that the current system may perform significantly better but a direct comparison is impossible without using the same corpus. It should also be considered that as both PatentEye and the system developed for this project rely on ChemicalTagger that improvements made to ChemicalTagger in the course of this project would likely also improve PatentEye's performance if it were updated.

The correct product was identified by PatentEye in 92% cases. The reason stated for the failures were false positive chemical entities in headings that could not be converted to structures. The fact that such "reactions" would always be rejected at the atom-mapping stage in the developed system further complicates comparison.

Correspondence with the authors of SCRIPDB indicated that the database included 190,083 reaction steps from USPTO patents for the period of 2008-2011. Of these only 7,873 possess a reaction arrow, a reagent and a product. As the source of the reactions is the CDX files rather than the text it would be potentially interesting to assess the level of overlap between these resources as a way of assessing how many reactions cannot be found from just the text. The results presently indicate that significantly more reactions can be extracted from the text but the number present in the CDX files may be understated due to reactions not being explicitly indicated as reactions and due to the use of generic structures to describe multiple reactions in one diagram.

4.10 Example use: solvent analysis

Besides obvious use cases, such as reaction searching, a large database of reactions allows one to start asking questions about the properties of the population of chemical reactions. For example, which are the most common solvents employed (Figure 4-14).

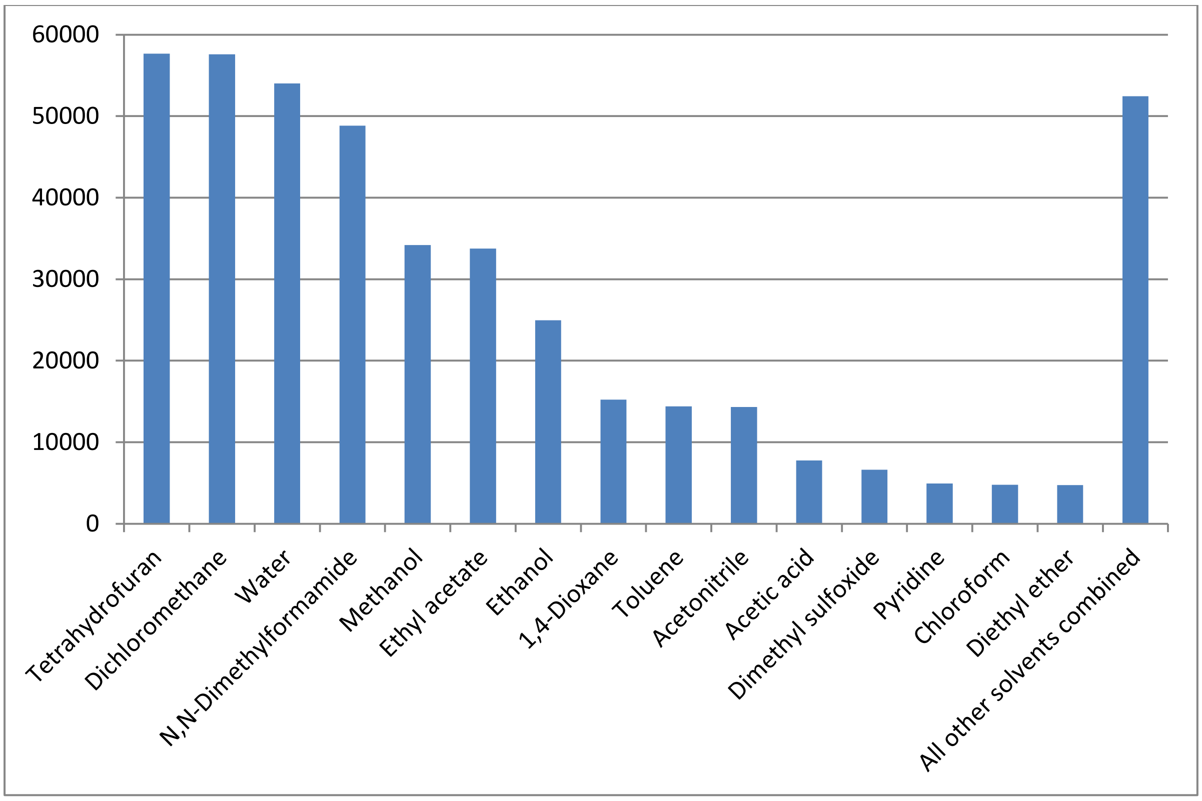


Figure 4-14 The top 15 solvents by frequency of occurrence in reactions

To produce Figure 4-14 unique solvent InChIs were recorded for each reaction. Where a single name indicates a mixture of solvents and hence produced one InChI this was split into its component InChIs using the heuristic that a mixture of solvents would be composed of neutral components.

One can observe that a few solvents occur disproportionally more than others. In total 627 discrete InChIs were detected indicating a potentially long tail. The sum of all solvents beyond the 15th is still less than the instances for any of the top 3 solvents indicating that most of these solvents are rarely used. A significant number of the InChIs that occurred very rarely are in fact not solvents so the figure of 627 for total solvents encountered is likely to be somewhat of an overestimate. Investigation of such cases could be useful for improving the precision of solvent detection.

With the growing importance of Green Chemistry[171], there is increased interest in finding alternatives to solvents, such as dichloromethane, that are known to have a negative environmental impact[172]. Being able to identify analogous reactions that were run in greener solvents is a potential use case.

4.11 Limitations and areas for future work

4.11.1 Interrelation between taggers

Conceptually, running a series of independent taggers is easy to understand and manipulate. However, to work ideally in practice, some taggers need the knowledge provided by other taggers. For example, the designation of words as chemicals and hence likely nouns would be useful to the POS tagger, since it was found to occasionally tag longer chemical entities as adjectives resulting in increased erroneous part of speech tag assignment to adjacent words.

Another example is in the regex tagger where certain words that are to be tagged may have a different part of speech depending on the context they are used in. This means that ideally the regex tagger should assign them different tags but as it has no knowledge of the context this is impossible necessitating the use of ad hoc post tagging tag corrections. If the regex tagger were aware of the tag assigned by the POS tagger this problem could often be resolved at the tagging step.

4.11.2 Chemical entity type assignment

Determiners in front of chemical names are used inconsistently. One would expect to be able to use the presence of 'a'/'an' to indicate that a chemical entity was describing a class of chemicals and to use 'the' to indicate that the chemical entity referred to a particular substance referenced previously. In practice, possibly due to not all patents being written by native speakers, the presence of a determiner is insufficient to rule out the interpretation that a chemical entity is of type "exact".

4.11.3 Solvents contained within another entity

In some cases the specification of the solvent is included through the use of a bracketed description of the solvent immediately after the solute chemical entity. ChemicalTagger will associate the bracketed description with the preceding solute entity rather than considering the solvent as a distinct chemical entity. As a result the solvent is not recorded in these cases. Another case where the solvent is not identified is where it is only implicitly described by an adjective e.g. 'aqueous' or 'methanolic'. If the meaning of the adjective is understood determining the solvent would be trivial.

4.11.4 Acid/Base workup steps

One of the leading causes of false positives was compounds that took part in an acid or base workup step that were not identified as being part of the workup. This could be partially addressed by allowing ChemicalTagger to identify the keyword ‘neutralise’ and hence identify neutralisation steps.

4.11.5 Additional roles

The role of desiccant should be added to describe the common use of compounds like sodium sulfate as drying agents. These compounds should be present in the list of spectator chemicals, but are neither catalysts nor solvents.

4.11.6 Structurally unknown intermediates

It is not uncommon in a multi-step reaction to have intermediate compounds. Often for brevity these compounds are referred to only by an important functional group e.g. ‘protected amine’ or even just as a description of the substance e.g. ‘grey powder’. Currently such reactions cannot be atom-mapped as the product of the first step of the reaction will have no structure. The same is true if this compound is used as a starting material in the next reaction.

4.11.7 Presentation of reactions

The results of the AAM could be used to align the depictions of the reactants and products making it clearer to see the transformation that has occurred.

4.11.8 Reaction conditions

Reaction conditions e.g. temperatures and the time taken for each step, are not currently extracted. Extracting such information would be a simple extension as the two exemplified properties are already appropriately tagged in ChemicalTagger’s output by the `TempPhrase` and `TimePhrase` elements. Capturing such information was not seen as a high priority as reaction searching is typically done by structure rather than by conditions and, at present, the extracted reactions are not aimed to be a replacement for the original text.

4.12 Conclusions

This work has shown that it is practical to use text mining to build a large reaction database from the publically available chemical literature, specifically patents, without human intervention. The extracted reactions (as reaction SMILES) are publically available from the project’s BitBucket page[173] as is the code to perform the reaction extraction. With the input of more patent documents the creation of a database of over a million reactions should be a relatively trivial undertaking. To the best of the author’s knowledge such a database would become the largest publically accessible reaction database. Such a resource could be extremely useful to both commercial and academic institutions, particularly when they are unable to access fee requiring systems such as Reaxys or SciFinder.

The development of the reaction extraction system has led to many improvements in ChemicalTagger, OSCAR4, the OPSIN Document Extractor and OPSIN. Especially in the case of ChemicalTagger, it is hoped that the improvements made will be of benefit to other users of these libraries.

Chapter 5 Overall Summary of Results and Conclusions

The increasing size of the chemical literature on the one hand creates problems in identifying informational resources, but on the other hand gives access to an ever increasing amount of information. To address both these issues, this work has centred on the development and validation of tools to text-mine literature for chemical information.

The development of the chemical name to structure algorithm OPSIN has been a key achievement. The system employs a regular grammar and corresponding automaton to facilitate tokenisation and parsing of chemical names. OPSIN was shown through examples and both artificial and real world benchmarks, to have high coverage and precision on organic chemical nomenclature, rivalling and often exceeding the commercial solutions tested. The algorithm is shown to be applicable to named entity recognition and has been successfully included into a frequently utilised public web service. Already, OPSIN itself appears to have achieved wide usage in the chemistry community, and as other comparable open-source solutions are not currently available is likely to increase in usage.

Building on the capabilities of OPSIN, in reliable and precise name to structure conversion, it was used as a critical component in the creation of a system for extracting chemical reactions from text-based literature. The system was demonstrated to be able to identify experimental sections and identify chemical entities. The entities are assigned roles, types and associated with quantities specified in the text. Finally atom-mapping is employed primarily to remove implausible reactions. The system was validated by inputting text from over 65,000 patent applications, and sampling the output of 424,621 extracted chemical reactions. This indicated a successful output in that 95% of them captured the essence of the reaction.

The extraction process is fast and requires little human intervention making it highly scalable. Hence the system could be used to facilitate much larger scale extraction of reactions from the patent literature. This would prove useful not only for the more obvious usage by synthetic chemists looking for routes of synthesis but also may prove useful in analysing trends in the chemistry used in organic syntheses. The system has the potential to benefit the community by allowing access to a large number of reactions without the restrictions or costs of traditional reaction searches. Publishers of organic chemistry journals may also be interested as a way of adding value to their articles by making them reaction searchable.

All of the software solutions developed as part of this project are open source and made freely available via BitBucket (*cf*. Appendix A). In this way these projects may be used in a complementary manner to other open source chemistry projects[73]. The potential also exists for them to be modified, improved and extended, in ways not necessarily conceived of by the author, allowing for wider usage than with traditionally more rigid commercial solutions. The software developed in this project is expected to prove useful to the cheminformatics community and, in the case of more user friendly services such as the OPSIN web service, the general chemistry community.

References

(1) Alexandru Dan Corlan. Medline trend: automated yearly statistics of PubMed results for any query. http://www.webcitation.org/65RkD48SV (accessed Feb 14, 2012).

(2) Economics and Statistics Division, WIPO. *World Intellectual Property Indicators, 2011 edition*; 2011; http://www.wipo.int/ipstats/en/statistics/patents/.

(3) Apache Software Foundation. Apache Lucene. http://lucene.apache.org/ (accessed Feb 21, 2012).

(4) Ashburner, M.; Ball, C. A.; Blake, J. A.; Botstein, D.; Butler, H.; Cherry, J. M.; Davis, A. P.; Dolinski, K.; Dwight, S. S.; Eppig, J. T.; Harris, M. A.; Hill, D. P.; Issel-Tarver, L.; Kasarskis, A.; Lewis, S.; Matese, J. C.; Richardson, J. E.; Ringwald, M.; Rubin, G. M.; Sherlock, G. Gene Ontology: tool for the unification of biology. *Nature Genetics* **2000**, *25*, 25–29.

(5) Degtyarenko, K.; de Matos, P.; Ennis, M.; Hastings, J.; Zbinden, M.; McNaught, A.; Alcantara, R.; Darsow, M.; Guedj, M.; Ashburner, M. ChEBI: a database and ontology for chemical entities of biological interest. *Nucl. Acids Res.* **2008**, *36*, D344–350.

(6) Schneider, G.; Fechner, U. Computer-based de novo design of drug-like molecules. *Nat Rev Drug Discov* **2005**, *4*, 649–663.

(7) Liu, H.; Hu, Z. Z.; Torii, M.; Wu, C.; Friedman, C. Quantitative assessment of dictionary-based protein named entity tagging. *J Am Med Inform Assoc* **2006**, *13*, 497–507.

(8) Wren, J. A scalable machine-learning approach to recognize chemical names within large text databases. *BMC Bioinformatics* **2006**, *7*, S3.

(9) Corbett, P.; Copestake, A. Cascaded classifiers for confidence-based chemical named entity recognition. *BMC Bioinformatics* **2008**, *9*, S4.

(10) Boudin, F.; Torres-Moreno, J.; El-Bèze, M. Mixing statistical and symbolic approaches for chemical names recognition. *Computational Linguistics and Intelligent Text Processing* **2008**, 334–343.

(11) Klinger, R.; Kolarik, C.; Fluck, J.; Hofmann-Apitius, M.; Friedrich, C. M. Detection of IUPAC and IUPAC-like chemical names. *Bioinformatics* **2008**, *24*, i268.

(12) Grego, T.; Pęzik, P.; Couto, F. M.; Rebholz-Schuhmann, D. Identification of Chemical Entities in Patent Documents. In *Proceedings of the 10th International Work-Conference on Artificial Neural Networks: Part II: Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*; IWANN ’09; Springer-Verlag: Berlin, Heidelberg, 2009; pp. 942–949.

(13) Sun, B.; Mitra, P.; Lee Giles, C.; Mueller, K. T. Identifying, Indexing, and Ranking Chemical Formulae and Chemical Names in Digital Documents. *ACM T Inform Syst* **2011**, *29*, 12.

(14) Jessop, D. M.; Adams, S.; Willighagen, E. L.; Hawizy, L.; Murray-Rust, P. OSCAR4: a flexible architecture for chemical text-mining. *J Cheminf* **2011**, 41.

(15) Rocktäschel, T.; Weidlich, M.; Leser, U. ChemSpot: a hybrid system for chemical named entity recognition. *Bioinformatics* **2012**, *28*, 1633–1640.

(16) Sayle, R.; Xie, P. H.; Muresan, S. Improved Chemical Text Mining of Patents with Infinite Dictionaries and Automatic Spelling Correction. *J. Chem. Inf. Model.* **2011**, *52*, 51–62.

(17) Park, J.; Rosania, G.; Shedden, K.; Nguyen, M.; Lyu, N.; Saitou, K. Automated extraction of chemical structure information from digital raster images. *Chem. Cent. J.* **2009**, *3*, 4.

(18) Filippov, I. V.; Nicklaus, M. C. Optical Structure Recognition Software To Recover Chemical Information: OSRA, An Open Source Solution. *J. Chem. Inf. Model.* **2009**, *49*, 740–743.

(19) Valko, A. T.; Johnson, A. P. CLiDE Pro: The Latest Generation of CLiDE, a Tool for Optical Chemical Structure Recognition. *J. Chem. Inf. Model.* **2009**, *49*, 780–787.

(20) Zimmermann, M. Chemical Structure Reconstruction with chemoCR. *TREC-CHEM 2011* **2011**.

(21) Smolov, V.; Zentsev, F.; Rybalkin, M. Imago: open-source toolkit for 2D chemical structure image recognition. *TREC-CHEM 2011* **2011**.

(22) Sadawi, N. M.; Sexton, A. P.; Sorge, V. Chemical Structure Recognition: A Rule Based Approach. In *19th Document Recognition and Retrieval Conference*; 2012.

(23) Fujiyoshi, A.; Nakagawa, K.; Suzuki, M. Robust Method of Segmentation and Recognition of Chemical Structure Images in ChemInfty. In *Pre-Proceedings of the 9th IAPR International Workshop on Graphics Recognition*; Seoul, South Korea, 2011.

(24) Lounnas, V.; Vriend, G. AsteriX: A Web Server To Automatically Extract Ligand Coordinates from Figures in PDF Articles. *J. Chem. Inf. Model.* **2012**, *52*, 568–576.

(25) Filippov, I. V.; Nicklaus, M. C.; Kinney, J. Improvements in Optical Structure Recognition Application. In *Document Analysis Systems Workshop*; Boston, 2010.

(26) Yan, S.; Spangler, W. S.; Chen, Y. Cross Media Entity Extraction and Linkage for Chemical Documents. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*; San Francisco, 2011.

(27) Van Noorden, R. Trouble at the text mine. *Nature* **2012**, *483*, 134–135.

(28) Peter Pappas, J. R. B. USPTO Teams with Google to Provide Bulk Patent and Trademark Data to the Public. http://www.uspto.gov/news/pr/2010/10_22.jsp (accessed Jun 4, 2012).

(29) Feng, C.; Yamashita, F.; Hashida, M. Automated Extraction of Information from the Literature on Chemical-CYP3A4 Interactions. *J. Chem. Inf. Model.* **2007**, *47*, 2449–2455.

(30) Yamashita, F.; Feng, C.; Yoshida, S.; Itoh, T.; Hashida, M. Automated Information Extraction and Structure−Activity Relationship Analysis of Cytochrome P450 Substrates. *J. Chem. Inf. Model.* **2011**, *51*, 378–385.

(31) Jiao, D.; Wild, D. J. Extraction of CYP Chemical Interactions from Biomedical Literature Using Natural Language Processing Methods. *J. Chem. Inf. Model.* **2009**, *49*, 263–269.

(32) Donaldson, I.; Martin, J.; de Bruijn, B.; Wolting, C.; Lay, V.; Tuekam, B.; Zhang, S.; Baskin, B.; Bader, G. D.; Michalickova, K.; Pawson, T.; Hogue, C. W. PreBIND and Textomy – mining the biomedical literature for protein-protein interactions using a support vector machine. *BMC Bioinformatics* **2003**, *4*, 11.

(33) Batchelor, C. R.; Corbett, P. T. Semantic enrichment of journal articles using chemical named entity recognition. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*; ACL ’07; Association for Computational Linguistics: Stroudsburg, PA, 2007; pp. 45–48.

(34) Swain, M. chemicalize.org. *J. Chem. Inf. Model.* **2012**, *52*, 613–615.

(35) Pafilis, E.; O’Donoghue, S. I.; Jensen, L. J.; Horn, H.; Kuhn, M.; Brown, N. P.; Schneider, R. Reflect: augmented browsing for the life scientist. *Nat Biotechnol.* **2009**, *27*, 508–510.

(36) Digital Science. SureChem. https://surechem.com/ (accessed Jun 4, 2012).

(37) Chen, Y.; Spangler, S.; Kreulen, J.; Boyer, S.; Griffin, T. D.; Alba, A.; Behal, A.; He, B.; Kato, L.; Lelescu, A.; Kieliszewski, C.; Wu, X.; Zhang, L. SIMPLE: a strategic information mining platform for licensing and execution. In *Proceedings of the 2009 IEEE International Conference on Data Mining Workshops*; 2009; pp. 270–275.

(38) Kayala, M. A.; Azencott, C.-A.; Chen, J. H.; Baldi, P. Learning to Predict Chemical Reactions. *J. Chem. Inf. Model.* **2011**, *51*, 2209–2222.

(39) Harold, E. R. XOM Design Principles. In *Proceedings of Extreme Markup Languages*; Montréal, Québec, 2004.

(40) Elliotte R. Harold. XOM. http://www.xom.nu/ (accessed Jun 4, 2012).

(41) Murray-Rust, P.; Rzepa, H. S. Chemical Markup, XML, and the Worldwide Web. 1. Basic Principles. *J. Chem. Inf. Comput. Sci.* **1999**, *39*, 928–942.

(42) Murray-Rust, P.; Rzepa, H. S. CML: Evolution and design. *J Cheminf* **2011**, *3*, 44.

(43) Murray-Rust, P.; Rzepa, H. S. Chemical Markup, XML, and the World Wide Web. 4. CML Schema. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 757–772.

(44) Chemical Markup Language Schema 3. http://www.xml-cml.org/schema/ (accessed Jun 4, 2012).

(45) Joe Townsend. Chemical Markup Language Validator. http://validator.xml-cml.org/ (accessed Jun 4, 2012).

(46) García, A.; Murray-Rust, P.; Wakelin, J. The use of XML and CML in computational chemistry and physics programs. In *Proceedings of the UK e-Science All Hands Meeting 2004*; 2004; pp. 1111–1114.

(47) Kuhn, S.; Helmus, T.; Lancashire, R. J.; Murray-Rust, P.; Rzepa, H. S.; Steinbeck, C.; Willighagen, E. L. Chemical Markup, XML, and the World Wide Web. 7. CMLSpect, an XML Vocabulary for Spectral Data. *J. Chem. Inf. Model.* **2007**, *47*, 2015–2034.

(48) Adams, N.; Winter, J.; Murray-Rust, P.; Rzepa, H. S. Chemical Markup, XML and the World-Wide Web. 8. Polymer Markup Language. *J. Chem. Inf. Model.* **2008**, *48*, 2118–2128.

(49) Holliday, G. L.; Murray-Rust, P.; Rzepa, H. S. Chemical Markup, XML, and the World Wide Web. 6. CMLReact, an XML Vocabulary for Chemical Reactions. *J. Chem. Inf. Model.* **2006**, *46*, 145–157.

(50) Weininger, D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 31–36.

(51) OpenSMILES Specification. http://www.opensmiles.org (accessed Jun 4, 2012).

(52) Daylight Chemical Information Systems. Daylight Theory: SMILES. http://www.daylight.com/dayhtml/doc/theory/theory.smiles.html (accessed Jun 4, 2012).

(53) Steinbeck, C.; Han, Y.; Kuhn, S.; Horlacher, O.; Luttmann, E.; Willighagen, E. The Chemistry Development Kit (CDK): An Open-Source Java Library for Chemo- and Bioinformatics. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 493–500.

(54) O'Boyle, N.; Banck, M.; James, C.; Morley, C.; Vandermeersch, T.; Hutchison, G. Open Babel: An open chemical toolbox. *J Cheminf* **2011**, *3*, 33.

(55) GGA Software Services. Indigo Toolkit. http://ggasoftware.com/opensource/indigo (accessed Jun 4, 2012).

(56) IUPAC. The IUPAC International Chemical Identifier (InChI). www.iupac.org/inchi/ (accessed Jun 4, 2012).

(57) Chomsky, N. Three models for the description of language. *IRE Trans. Inf. Theory* **1956**, *2*, 113–124.

(58) Adams, S. E.; Goodman, J. M.; Kidd, R. J.; McNaught, A. D.; Murray-Rust, P.; Norton, F. R.; Townsend, J. A.; Waudby, C. A. Experimental data checker: better information for organic chemists. *Org. Biomol. Chem.* **2004**, *2*, 3067.

(59) Townsend, J. A.; Adams, S. E.; Waudby, C. A.; de Souza, V. K.; Goodman, J. M.; Murray-Rust, P. Chemical documents: machine understanding and automated information extraction. *Org. Biomol. Chem.* **2004**, *2*, 3294.

(60) Corbett, P.; Murray-Rust, P. High-Throughput Identification of Chemistry in Life Science Texts. *Lecture Notes in Comput. Sci.* **2006**, *4216*, 107–118.

(61) Hawizy, L.; Jessop, D. M.; Adams, N.; Murray-Rust, P. ChemicalTagger: A tool for semantic text-mining in chemistry. *J Cheminf* **2011**, *3*, 17.

(62) Apache OpenNLP. http://incubator.apache.org/opennlp/ (accessed Jun 4, 2012).

(63) Marcus, M. P.; Marcinkiewicz, M. A.; Santorini, B. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics* **1993**, *19*, 313–330.

(64) Parr, T. *The definitive ANTLR reference: building domain-specific languages*; Pragmatic Bookshelf, 2007.

(65) Hearst, M. A. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th International Conference on Computational Linguistics: Volume 2*; 1992; pp. 539–545.

(66) Apache Maven Home Page. http://maven.apache.org/ (accessed Jun 4, 2012).

(67) Atlassian. Bitbucket. https://bitbucket.org/ (accessed Jun 4, 2012).

(68) GitHub. https://github.com/ (accessed Jun 4, 2012).

(69) Mercurial SCM. http://mercurial.selenic.com/ (accessed Jun 4, 2012).

(70) JUnit. http://www.junit.org/ (accessed Jun 4, 2012).

(71) Jenkins. http://jenkins-ci.org/ (accessed Jun 4, 2012).

(72) Lowe, D. M.; Corbett, P. T.; Murray-Rust, P.; Glen, R. C. Chemical Name to Structure: OPSIN, an Open Source Solution. *J. Chem. Inf. Model.* **2011**, *51*, 739–753.

(73) O'Boyle, N. M.; Guha, R.; Willighagen, E. L.; Adams, S. E.; Alvarsson, J.; Bradley, J.-C.; Filippov, I. V.; Hanson, R. M.; Hanwell, M. D.; Hutchison, G. R.; James, C. A.; Jeliazkova, N.; Lang, A. S.; Langner, K. M.; Lonie, D. C.; Lowe, D. M.; Pansanel, J.; Pavlov, D.; Spjuth, O.; Steinbeck, C.; Tenderholt, A. L.; Theisen, K. J.; Murray-Rust, P. Open Data, Open Source and Open Standards in chemistry: The Blue Obelisk five years on. *J Cheminf 3*, 37.

(74) Pictet, A. Le Congrès International de Genève pour la Réforme de la Nomenclature Chimique. Archives des sciences physiques et naturelles: Geneva, 1892; Vol. 27, pp. 485–520.

(75) Definitive Rules for Nomenclature of Organic Chemistry. *J. Am. Chem. Soc.* **1960**, *82*, 5545–5574.

(76) IUPAC. *Nomenclature of Organic Chemistry*; Pergamon Press, Oxford, 1979.

(77) IUPAC. *A Guide to IUPAC Nomenclature of Organic Compounds (Recommendations 1993)*; Blackwell Scientific publications, 1993.

(78) IUPAC. Draft Nomenclature of Organic Chemistry. http://old.iupac.org/reports/provisional/abstract04/favre_310305.html (accessed Jun 4, 2012).

(79) *Nomenclature of Inorganic Chemistry: Recommendations 1990*; Blackwell Scientific Publications, 1990.

(80) *Nomenclature of Inorganic Chemistry: IUPAC Recommendations 2005*; Cambridge, UK: Royal Society of Chemistry Publishing/IUPAC, 2005.

(81) Smith, H. A. The Centennial of Systematic Organic Nomenclature. *J. Chem. Educ.* **1992**, *69*, 863.

(82) Donaldson, N.; Powell, W. H.; Rowlett, R. J.; White, R. W.; Yorka, K. V. Chemical Abstracts Index Names for Chemical Substances in the Ninth Collective Period. *J. Chem. Doc.* **1974**, *14*, 3–15.

(83) Chemical Abstracts Service. Naming and Indexing of Chemical Substances for Chemical Abstracts. *American Chemical Society* **2007**.

(84) Garfield, E. An Algorithm for Translating Chemical Names to Molecular Formulas. *J. Chem. Doc.* **1962**, *2*, 177–179.

(85) Garfield, E. An Algorithm for Translating Chemical Names to Molecular Formulas. *Essays of an Information Scientist* **1984**, *7*, 441–513.

(86) Vander Stouw, G. G.; Naznitsky, I.; Rush, J. E. Procedures for Converting Systematic Names of Organic Compounds into Atom-Bond Connection Tables. *J. Chem. Doc.* **1967**, *7*, 165–169.

(87) Vander Stouw, G. G.; Elliott, P. M.; Isenberg, A. C. Automated Conversion of Chemical Substance Names to Atom-Bond Connection Tables. *J. Chem. Doc.* **1974**, *14*, 185–193.

(88) Cooke-Fox, D. I.; Kirby, G. H.; Rayner, J. D. Computer translation of IUPAC systematic organic chemical nomenclature. 1. Introduction and background to a grammar-based approach. *J. Chem. Inf. Comput. Sci.* **1989**, *29*, 101–105.

(89) Cooke-Fox, D. I.; Kirby, G. H.; Rayner, J. D. Computer translation of IUPAC systematic organic chemical nomenclature. 2. Development of a formal grammar. *J. Chem. Inf. Comput. Sci.* **1989**, *29*, 106–112.

(90) Cooke-Fox, D. I.; Kirby, G. H.; Rayner, J. D. Computer translation of IUPAC systematic organic chemical nomenclature. 3. Syntax analysis and semantic processing. *J. Chem. Inf. Comput. Sci.* **1989**, *29*, 112–118.

(91) Cooke-Fox, D. I.; Kirby, G. H.; Lord, M. R.; Rayner, J. D. Computer translation of IUPAC systematic organic chemical nomenclature. 4. Concise connection tables to structure diagrams. *J. Chem. Inf. Comput. Sci.* **1990**, *30*, 122–127.

(92) Cooke-Fox, D. I.; Kirby, G. H.; Lord, M. R.; Rayner, J. D. Computer translation of IUPAC systematic organic chemical nomenclature. 5. Steroid nomenclature. *J. Chem. Inf. Comput. Sci.* **1990**, *30*, 128–132.

(93) Kirby, G. H.; Lord, M. R.; Rayner, J. D. Computer translation of IUPAC systematic organic chemical nomenclature. 6.(Semi) automatic name correction. *J. Chem. Inf. Comput. Sci.* **1991**, *31*, 153–160.

(94) Ikutoshi Matsuura. Development of a System for Translation of Chemical Name into 2D-Structure (V). *26th Symposium on Chemical Information and Computer Science* **2003**, 101–104.

(95) Ikutoshi Matsuura. Development of a System for Translation of Chemical Name into 2D-Structure (VI). *27th Symposium on Chemical Information and Computer Science* **2004**, 63–66.

(96) Ikutoshi Matsuura. Development of a System for Translation of Chemical Name into 2D-Structure (VII). *28th Symposium on Chemical Information and Computer Science* **2005**, 29–32.

(97) University of Manchester. ChemNomParse. http://chemnomparse.sourceforge.net/ (accessed Jun 4, 2012).

(98) Banville, D. L. *Chemical Information Mining: Facilitating Literature-Based Discovery*; 1st ed.; CRC Press, 2008.

(99) *ACD/Name*; ACD/Labs: Toronto, Canada; http://www.acdlabs.com/.

(100) Bio-Rad Laboratories. *IUPAC DrawIt*; Hercules, CA; http://www.bio-rad.com.

(101) *Struct=Name*; PerkinElmer: Cambridge, MA; http://www.cambridgesoft.com.

(102) *Name to structure*; ChemAxon: Budapest, Hungary; http://www.chemaxon.com/.

(103) *NameExpert*; ChemInnovation Software: San Diego, CA; http://www.cheminnovation.com.

(104) *Name to structure*; InfoChem: Munich, Germany; http://infochem.de/.

(105) *Lexichem ToolKit*; OpenEye Scientific Software: Santa Fe, NM; http://www.eyesopen.com.

(106) Brecher, J. Name=Struct: A Practical Approach to the Sorry State of Real-Life Chemical Nomenclature. *J. Chem. Inf. Comput. Sci.* **1999**, *39*, 943–950.

(107) Brecher, J. S. Method, system, and software for deriving chemical structural information. US Patent 7,054,754, May 30, 2006.

(108) Lawson, A. J.; Roller, S.; Grotz, H.; Wisniewski, J. L.; Kelkheim, L. G. Method and software for extracting chemical data. EPO Patent EP20050252713, November 1, 2006.

(109) Engelken, H. A System for Semantic Analysis of Chemical Compound Names. In *Proceedings of the ACL-IJCNLP 2009 Student Research Workshop*; Suntec, Singapore, 2009; pp. 36–44.

(110) Møller, A. dk.brics.automaton – Finite-State Automata and Regular Expressions for Java. http://www.brics.dk/automaton/ (accessed Jun 4, 2012).

(111) Jensen, W. B. A Quantitative van Arkel Diagram. *J. Chem. Educ.* **1995**, *72*, 395.

(112) Lozac'h, N. Extension of Rules A-1.1 and A-2.5 concerning numerical terms used in organic chemical nomenclature (Recommendations 1986). *Pure Appl. Chem.* **1986**, *58*, 1693–1696.

(113) Moss, G. P. Extension and revision of the von Baeyer system for naming polycyclic compounds (including bicyclic compounds). *Pure Appl. Chem.* **1999**, *71*, 513–529.

(114) Moss, G. P. Extension and revision of the nomenclature for spiro compounds. *Pure Appl. Chem.* **1999**, *71*, 531–558.

(115) Moss, G. P. Nomenclature of Fused and Bridged Fused Ring Systems (IUPAC Recommendations 1998). *Pure Appl. Chem.* **1998**, *70*, 143–216.

(116) Powell, W. Revision of the Extended Hantzsch-Widman System of Nomenclature for Heteromonocycles. *Pure Appl. Chem.* **1983**, *55*, 409–416.

(117) Powell, W. Treatment of Variable Valence in Organic Nomenclature (Lambda Convention). *Pure Appl. Chem.* **1984**, *56*, 769–778.

(118) Nomenclature and symbolism for amino acids and peptides (Recommendations 1983). *Pure Appl. Chem.* **1984**, *56*, 595–624.

(119) McNaught, A. D. Nomenclature of carbohydrates (IUPAC Recommendations 1996). *Pure Appl. Chem.* **1996**, *68*, 1919–2008.

(120) Kahovec, J.; Fox, R. B.; Hatada, K. Nomenclature of regular single-strand organic polymers (IUPAC Recommendations 2002). *Pure Appl. Chem.* **2002**, *74*, 1921–1956.

(121) Tchekhovskoi, D. InChI Canonicalization Algorithm. http://sourceforge.net/mailarchive/forum.php?thread_name=5.1.1.5.2.20050708111329.02502190%40email.nist.gov&forum_name=inchi-discuss (accessed Jun 4, 2012).

(122) InChI Technical Manual (Version 1.04). http://www.inchi-trust.org/downloads/ (accessed Jun 4, 2012).

(123) Cahn, R. S.; Ingold, C.; Prelog, V. Specification of molecular chirality. *Angew. Chem., Int. Ed. Engl.* **1966**, *5*, 385–415.

(124) Prelog, V.; Helmchen, G. Basic Principles of the CIP-System and Proposals for a Revision. *Angew. Chem., Int. Ed. Engl.* **1982**, *21*, 567–583.

(125) Mata, P.; Lobo, A. M.; Marshall, C.; Johnson, A. P. The CIP sequence rules: Analysis and proposal for a revision. *Tetrahedron: Asymmetry* **1993**, *4*, 657–668.

(126) Razinger, M.; Balasubramanian, K.; Perdih, M.; Munk, M. E. Stereoisomer generation in computer-enhanced structure elucidation. *J. Chem. Inf. Comput. Sci.* **1993**, *33*, 812–825.

(127) Giles, P. M. Revised Section F: Natural products and related compounds. *Pure Appl. Chem.* **1999**, *71*, 587–643.

(128) Sam Adams. JNI-InChI. http://jni-inchi.sourceforge.net/ (accessed Jun 4, 2012).

(129) Eller, G. A. Improving the Quality of Published Chemical names with Nomenclature Software. *Molecules* **2006**, *11*, 915–28.

(130) O'Boyle, N. M.; Morley, C.; Hutchison, G. R. Pybel: a Python wrapper for the OpenBabel cheminformatics toolkit. *Chemistry Central Journal* **2008**, *2*.

(131) USPTO Patent Application Publication Full Text with Embedded Images. http://www.google.com/googlebooks/uspto-patents-applications-text-with-embedded-images.html (accessed Jun 4, 2012).

(132) OPSIN Source Code on Bitbucket. http://bitbucket.org/dan2097/opsin/ (accessed Jun 4, 2012).

(133) Lowe, D. M. OPSIN Web Service. http://opsin.ch.cam.ac.uk/ (accessed Jun 4, 2012).

(134) Murray-Rust, P.; Townsend, J.; Downing, J.; Dirks, L.; Wade, A.; Naim, O.; Galos, M.; Haughton, T. Chemistry Add-in for Word - Microsoft Research. http://research.microsoft.com/chem4word/ (accessed Jun 4, 2012).

(135) Southan, C. Synergies between ChemAxon's chemicalize and other open resources to extract structures from patents, discern SAR, and find intersects or similarities in PubChem. Chemaxon UGM, Budapest, Hungary, May 23, 2012.

(136) Lowe, D. M. OPSIN Document Extractor. https://bitbucket.org/dan2097/opsin-document-extractor (accessed Jun 4, 2012).

(137) Tomoyuki, S. English to Japanese trivial chemical names. http://homepage1.nifty.com/nomenclator/triv/trivial.htm (accessed Jun 4, 2012).

(138) Williams, A. J.; Ekins, S. A quality alert and call for improved curation of public chemistry databases. *Drug Discovery Today* **2011**, *16*, 747–750.

(139) Clark, A. M. Accurate Specification of Molecular Structures: The Case for Zero-Order Bonds and Explicit Hydrogen Counting. *J. Chem. Inf. Model.* **2011**, *51*, 3149–3157.

(140) Damerau, F. J. A Technique for Computer Detection and Correction of Spelling Errors. *Communications of the ACM* **1964**, *7*, 171–176.

(141) Sayle, R. Foreign Language Translation of Chemical Nomenclature by Computer. *J. Chem. Inf. Model.* **2009**, *49*, 519–530.

(142) Jeliazkova, N.; Jeliazkov, V. AMBIT RESTful web services: an implementation of the OpenTox application programming interface. *J Cheminf* **2011**, *3*, 18.

(143) O'Boyle, N. M.; Hutchison, G. R. Cinfony–combining Open Source cheminformatics toolkits behind a common interface. *Chemistry Central Journal* **2008**, *2*, 24.

(144) Sitzmann, M. NCI/CADD Chemical Identifier Resolver. http://cactus.nci.nih.gov/chemical/structure (accessed Jun 4, 2012).

(145) Willighagen, E. OPSIN used for a Bioclipse wizard. http://chem-bla-ics.blogspot.com/2011/02/opsin-used-for-bioclipse-wizard.html (accessed Jun 4, 2012).

(146) Lawson, K. R.; Lawson, J. LICSS–A chemical spreadsheet in Microsoft Excel. *Journal of Cheminformatics* **2012**, *4*, 3.

(147) Weber, L. Chemical Ontologies for Life Sciences. Chemaxon UGM, Budapest, Hungary, May 23, 2012.

(148) International Union of Crystallography; ChemAxon. International Union of Crystallography chooses ChemAxon Name to Structure technology. http://www.chemaxon.com/news/international-union-of-crystallography-chooses-chemaxon-name-to-structure-technology/ (accessed Jun 4, 2012).

(149) Kinney, J. Validation and characterization of chemical structures derived from names and images in scientific documents. 9th International Conference on Chemical Structures, Noordwijkerhout, The Netherlands, June 7, 2011.

(150) Muresan, S. Automated spelling correction to improve recall rates of name-to-structure tools for chemical text mining. Chemaxon UGM, Budapest, Hungary, May 17, 2011.

(151) OPSIN used for generating SMILES from extracted chemical names, Personal communication from IBM **2011**.

(152) Blake, J. E.; Dana, R. C. CASREACT: more than a million reactions. *J. Chem. Inf. Comput. Sci.* **1990**, *30*, 394–399.

(153) Chemical Abstracts Service. CAS Databases - CASREACT, Chemical Reactions. http://www.cas.org/expertise/cascontent/casreact.html (accessed Jun 4, 2012).

(154) Goodman, J. Computer Software Review: Reaxys. *J. Chem. Inf. Model.* **2009**, *49*, 2897–2898.

(155) Elsevier Properties SA. Reaxys. https://www.reaxys.com/info/ (accessed Jun 4, 2012).

(156) Roth, D. L. SPRESIweb 2.1, a Selective Chemical Synthesis and Reaction Database. *J. Chem. Inf. Model.* **2005**, *45*, 1470–1473.

(157) InfoChem. SPRESIweb. http://www.spresi.com/ (accessed Jun 4, 2012).

(158) Thomson Reuters. Current Chemical Reactions. http://thomsonreuters.com/products_services/science/science_products/a-z/current_chemical_reactions/ (accessed Jun 4, 2012).

(159) Thieme Chemistry. Science of Synthesis. http://www.science-of-synthesis.com/en/products/reference-works/science-of-synthesis.html (accessed Jun 4, 2012).

(160) Wife, D. SORD (Selected Organic Reactions Database). http://www.sord.nl/ (accessed Jun 4, 2012).

(161) Organic Syntheses. http://www.orgsyn.org/ (accessed Jun 4, 2012).

(162) WebReactions. http://www.openmolecules.org/webreactions/ (accessed Jun 4, 2012).

(163) Lowe, D. M. Automated Extraction of Reactions from the Patent Literature. CINF#75, 243rd ACS National Meeting & Exposition, San Diego, CA, March 27, 2012.

(164) Reeker, L. H.; Zamora, E. M.; Blower, P. E. Specialized information extraction: automatic chemical reaction coding from English descriptions. In *Proceedings of the first conference on Applied natural language processing*; Association for Computational Linguistics, 1983; pp. 109–116.

(165) Zamora, E. M.; Blower Jr, P. E. Extraction of chemical reaction information from primary journal text using computational linguistics techniques. 1. Lexical and syntactic phases. *J. Chem. Inf. Comput. Sci.* **1984**, *24*, 176–181.

(166) Zamora, E. M.; Blower Jr, P. E. Extraction of chemical reaction information from primary journal text using computational linguistics techniques. 2. Semantic phase. *J. Chem. Inf. Comput. Sci.* **1984**, *24*, 181–188.

(167) Ai, C. S.; Blower Jr, P. E.; Ledwith, R. H. Extraction of chemical reaction information from primary journal text. *J. Chem. Inf. Comput. Sci.* **1990**, *30*, 163–169.

(168) Jessop, D. M.; Adams, S. E.; Murray-Rust, P. Mining Chemical Information from Open Patents. *Journal of Cheminformatics* **2011**, *3*, 40.

(169) Heifets, A.; Jurisica, I. SCRIPDB: a portal for easy access to syntheses, chemicals and reactions in patents. *Nucl. Acids Res.* **2011**, *40*, D428–D433.

(170) Jessop, D. M. Information extraction from chemical patents. Ph.D, University of Cambridge, 2011.

(171) Dunn, P. J. The importance of Green Chemistry in Process Research and Development. *Chemical Society Reviews* **2012**, *41*, 1452.

(172) Hargreaves, C. R.; Manley, J. B. Collaboration to Deliver a Solvent Selection Guide for the Pharmaceutical Industry. In *ACS GCI Pharmaceutical Roundtable*; ACS Green Chemistry Institute, 2008.

(173) Lowe, D. M. Patent Reaction Extraction Project. https://bitbucket.org/dan2097/patent-reaction-extraction (accessed Jun 4, 2012).

Appendix A

This project has resulted in the creation of a significant amount of code. For posterity the versions of the software that were current at the point of writing this thesis are attached as supporting information. For all projects both source code and binaries (inclusive of dependencies) are included in the /code directory. Note that only the OPSIN binary is executable, the other projects are exclusively used as libraries.

Software components developed for this project:

- OPSIN (version 1.2.0)
- OPSIN Document Extractor (version 1.0.1)
- OPSIN-ws (13th March 2012)
- Patent Reaction Extraction (version 1.0)

Newer versions of these software projects may be available from https://bitbucket.org/dan2097

The Patent Reaction Extraction code depends heavily on the improved versions of ChemicalTagger and OSCAR4 that were developed for this project:

- ChemicalTagger (version 1.3.1)
- OSCAR4 (version 4.1)

Newer versions of these software projects may be available from https://bitbucket.org/wwmm

Appendix B

To allow reproduction of the results in this thesis, both the data sets and results are included as supporting information for all cases where the size of the data did not make this impractical.

Chemical name to structure testing:

- /data/nameToStructure/Pubchem30000_dec2011 - Includes the Pubchem IDs, their corresponding SMILES and InChIs, the names generated from ACD/Name, ChemBioDraw, Lexichem and Marvin, and the InChIs generated by ChemBioDraw, Marvin and OPSIN on these names. A summary of the results which was used to generate Figure 3-143, Figure 3-144, Figure 3-145 and Figure 3-146 is included in 30000Pubchem_Dec11.xls.
- /data/nameToStructure/2011_oscar4_patentnames – Includes the chemical names found by OSCAR4 in 2011 organic chemistry patent application headings, the names after processing by OPSIN's pre-processor and the SMILES generates from both sets of names from ChemBioDraw, Marvin and OPSIN. These results were used to generate Figure 3-147.
- /data/nameToStructure/ChebiDec11 – Includes names and corresponding SMILES from compounds in the ChEBI database in December 2011. These names were used as the input to produce the results for Figure 3-9.

Reaction extraction testing:

- /data/reactionExtraction/evaluation – Includes the automatically extracted reactions randomly chosen for evaluation and the manual evaluation performed to test their quality. A breakdown of the number of patent applications with a certain number of reactions is also included as was used to generate Figure 4-13.
- /data/reactionExtraction/solvents – Includes the complete list of solvents and their occurrence counts. This was used to generate Figure 4-14.

Due to size constraints including the 2008-2011 organic chemistry patent applications is not practical but the ExtractOrganicChemistryPatents class used to filter patents to those containing IPC code C07 is included in the Patent Reaction Extraction project.

Appendix C

Terms in OPSIN’s grammar:

Grammar Symbol Description	Examples
a	An 'a'
acetalClass	acetal, ketal, hemiacetal, hemiketal
acidStem	acet, valer, succin
alkaneStemHundreds	hect, trict
alkaneStemModifier	iso, neo, tert
alkaneStemTens	dec, cos, icos
alkaneStemThousands	killi, dili
alkaneStemTrivial	meth, undec
alkaneStemUnits	hen, do, tri
alphaBetaStereochemLocant	3beta
amineMeaningNitrilo	amine as a substituent in the middle of a name
aminoAcidEndsInAn	tryptoph
aminoAcidEndsInIc	glutam, aspart
aminoAcidEndsInIne	lys, alan, glutam
aminoAcidYl	yl as in glycin-2-yl
ane	ane as in the ending of an alkane or heteroatom analogue
anhydrideFunctionalGroup	anhydride, peroxyanhydride
annulen	[8]annulen
basicFunctionalClass	ester, glycol, cyanohydrin
benzo	benzo as in benzo as a fused ring component
bigCapitalH	5H-
bridgeFormingO	'o' as in ethano
canBeDlPrefixedSimpleGroup	glucose, galactosamine
carbohydrateChain	triose, hexose
carbohydrateConfigurationalPrefix	glycero, gluco, manno
carbohydrateRingSize	oxirose, furanose, pyranose
carbohydrateStem	gluco, manno, fructo
chalcogenAcid	sulfon, sulfin, tellur
chalcogenReplacement	thio, seleno, telluro
chargeOrOxidationNumberSpecifier	(IV), (2+)
closeBracket	], },)
colonSeperatedLocant	1,2:3,4
comma	A comma that is ignored after parsing
cyclicUnsaturableHydrocarbon	menth, prism, adamant
cyclo	cyclo as in cyclopropane
dispiroter	1,2':7',2''-dispiroter
divalentFunctionalGroup	ketone, sulfone
dlStereochemistry	D-, L-, Dg-
e	An optional 'e'
elementaryAtom	sodium, natrium, zirconium
elidedAMultiplier	tetr, pent

endOfFunctionalGroup	Indicates the end of a functional group has been reached
endOfMainGroup	Indicates the end of the principal group
endOfSubstituent	Indicates the end of a substituent has been reached
epoxy	epoxy, epithio
FR2hydrocarbonComponent	cen, len, helicen
functionalModifier	poly
fusionBracket	[4,5-d], [3',4':5,6]
fusionRing	indolo, pyrido, pyrrolo
fusionRingAcceptsFrontLocants	naphthyridino, phenanthrolino
groupMultiplier	bis, tris, tetrakis
groupStemAllowingAllSuffixes	hydrazin
groupStemAllowingInlineSuffixes	amid, keten, formazan
hantzschWidmanSuffix	iran, olan, inan
heteroAtom	aza, azonia, azanylia, azanida
heteroAtomaElided	az, thi
heteroStem	alum, bor, oxid, sulf
hwAne	ane as in the ending of a Hantzsch-Widman system
hwAneCompatible	oxa, ox, thi
hwHeteroAtom	aza, arsa, bisma
hwIne	ine as in the ending of a Hantzsch-Widman system
hwIneCompatible	oxa, thia, selena
hydro	hydro, dehydro
hyphen	An optional hyphen
implicitIc	Added after unsuffixed amino acids to simplify systematic construction
ine	ine as in the ine of glycine
infixableInlineSuffix	oyl
inlineChargeSuffix	ium, ylium, ide, uide
inlineSuffix	yl, ylidyne, oyl, sulfonyl
inlineSuffixAllowingPrefixes	amido, oyl
interSubstituentHyphen	A hyphen between two substituents
lambdaConvention	3lambda5
lightRotation	(+), (-), (+-)
locant	2, S-, alpha, N5
locantThatNeedsBrackets	Subset of locantGroup
mono	mono as in locanted monophosphate
monoNuclearNonCarbonAcid	sulfam, azin, phosphon
monovalentFunctionalGroup	alcohol, thiol
multipleFusor	[2',3':3,4;2",3":6,7]
multiplyableFunctionalClass	oxime, oxide
naturalProductRequiresUnsaturator	morphin, androst
nitrogenHeteroStem	az as in diazano
nonCarbonAcidNoAcyl	diphosphon, boron, selen
o	An optional euphonic o

oMeaningYl	o as in glycino
openBracket	[, {, (
optionalCloseBracket	Same as closeBracket but ignored after parsing
optionalOpenBracket	Same as openBracket but ignored after parsing
orthoMetaPara	ortho, meta, para, o-, m-, p-
perhydro	perhydro
relativeCisTrans	r-5,c-5,t-7
repeatableInlineSuffix	yl, ylidene
replacementInfix	thi, perox, hydrazid, hydrazon
ringAssemblyMultiplier	bi, ter, quarter
simpleCyclicGroup	perbenzoic acid
simpleGroup	hydroxide, chloroform, thiuram disulfide
simpleGroupClass	amine, carboxylic acid (groups that must be substituted to not be generic)
simpleMultiplier	di, tri, tetra
simpleSubstituent	chloro, hydroxy, amino
spiro	spiro as in a polycyclic spiro system
spiroDescriptor	spiro[2.2]
spiroLocant	Subset of locantGroup used between components of a spiro system
spiroOldMethod	spiro as in a polycyclic spiro system (deprecated naming system)
standaloneMonovalentFunctionalGroup	chloride, cyanide
stereochemistryBracket	(2R), (2E,4Z)
structuralCloseBracket	Same as closeBracket but used to assist in nomenclature interpretation
structuralOpenBracket	Same as openBracket but used to assist in nomenclature interpretation
subtractivePrefix	deoxy, desoxy
suffix	one, ol, carboxylic acid
suffixableSubstituent	sil, vin
suffixesThatCanBeModifiedByAPrefix	amide, ate
suffixPrefix	sulfon, sulfin, carbono
symPolycyclicSpiro	spirobi, spiroter
trivialRing	benzen, pyridin, toluen
trivialRingSubstituent	phenyl
trivialRingSubstituentAnySuffix	pyrid, acrid
trivialRingSubstituentInlineOnly	imidaz, tol
unbrackettedStereochem	E, trans
unsaturator	ene, en, yne
vonBaeyer	cyclo[2.2.2]
vonBaeyerMultiplier	bi, tri, tetra
ylamine	ylamine (used in conjunctive nomenclature)
ylene	ylene