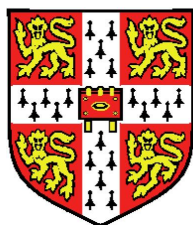


CHEMINFORMATICS FOR GENOME-SCALE METABOLIC RECONSTRUCTIONS



John W. May

European Molecular Biology Laboratory
European Bioinformatics Institute

University of Cambridge
Homerton College

A thesis submitted for the degree of

Doctor of Philosophy

June 2014

Declaration

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

This dissertation is not substantially the same as any I have submitted for a degree, diploma or other qualification at any other university, and no part has already been, or is currently being submitted for any degree, diploma or other qualification.

This dissertation does not exceed the specified length limit of 60,000 words as defined by the Biology Degree Committee. This dissertation has been typeset using \LaTeX in 11 pt Palatino, one and half spaced, according to the specifications defined by the Board of Graduate Studies and the Biology Degree Committee.

June 2014

John W. May

to Róisín

Acknowledgements

This work was carried out in the Cheminformatics and Metabolism Group at the European Bioinformatics Institute (EMBL-EBI). The project was funded by Unilever, the Biotechnology and Biological Sciences Research Council [BB/I532153/1], and the European Molecular Biology Laboratory.

I would like to thank my supervisor, Christoph Steinbeck for his guidance and providing intellectual freedom. I am also thankful to each member of my thesis advisory committee: Gordon James, Julio Saez-Rodriguez, Kiran Patil, and Gos Micklem who gave their time, advice, and guidance.

I am thankful to all members of the Cheminformatics and Metabolism Group. In particular, the extremely knowledgeable ChEBI curators as well as past and present members of the research team: Pablo Moreno, Stephan Beisken, Luís de Figueiredo, Felicity Allen, and Kalai Vanii Jayaseelan.

I owe a great deal of thanks to Egon Willighagen for meticulously reviewing the deluge of patches, programming advice, and for encouraging me to start a blog.

I am grateful to attendees of the Cambridge Cheminformatics Network Meetings for stimulating conversations. Specifically, I would like to thank Roger Sayle for many enjoyable discussions on efficient algorithms and for prompting me to address some long standing issues in the CDK library. I am also thankful to Mark Vine for valuable input and advice.

I would like to thank: Gordon James, Stephan Beisken, Pablo Moreno, Egon Willighagen, and Róisín Sarsfield for reading and providing feedback of draft chapters of this dissertation.

Finally, on a personal note, I would like to thank my loving parents, my sister Alexandra, and my partner Róisín who have always encouraged me to pursue what I find interesting and for providing support throughout my studies.

Abstract

Genome-scale metabolic reconstructions are an important resource in the study of metabolism. They provide both a system and component level view of the biochemical transformations of metabolites. As more reconstructions have been created it remains a challenge to integrate and reason about their contents. This thesis focuses on the development of computational methods to allow on-demand comparison and alignment of metabolic reconstructions.

A novel method is introduced that utilises chemical structure representations to identify equivalent metabolites between reconstructions. Using a graph theoretic representation allows the identification and reasoning of metabolites that have a non-exact match. A key advantage is that the method uses the contents of reconstructions directly and does not rely on the creation or use of a common reference.

To annotate reconstructions with chemical structure representations an interactive desktop application is introduced. The application assists in the creation and curation of metabolic information using manual, semi-automated, and automated methods. Chemical structure representations can be retrieved, drawn, or generated to allow precise metabolite annotation.

In processing chemical information, efficient and optimised algorithms are required. Several areas are addressed and implementations have been contributed to the Chemistry Development Kit. Rings are a fundamental property of chemical structures therefore multiple ring definitions and fast algorithms are explored. Conversion and standardisation between structure representations present a challenge. Efficient algorithms to determine aromaticity, assign a Kekulé form, and generate tautomers are detailed.

Many enzymes are selective and specific to stereochemistry. Methods for the identification, depiction, comparison, and description of stereochemistry are described.

Contents

Contents	xi
List of Figures	xix
1 Introduction	1
1.1 Introduction to genome-scale metabolic models	2
1.1.1 Constraint-based analysis	2
1.1.2 Creation	3
1.1.3 Exchange	5
1.1.4 Databases	6
1.1.5 Software tools	8
1.2 Introduction to cheminformatics	12
1.2.1 Representation and applications	12
1.2.2 Exchange	14
1.2.3 Software libraries	16
1.3 Chemical information in metabolic reconstructions	18
1.3.1 Applications	18
1.3.2 Availability of reconstructions with chemical structures	21
1.4 Objectives of this thesis	23
1.5 Publications	24
I Chemical data in metabolic reconstructions	25
2 Alignment of reconstructions	27
2.1 Introduction	27
2.1.1 Motivation	27

2.1.2	Identifying metabolites	27
2.1.3	Existing approaches	29
2.1.4	Introduction to chemical identity	30
2.1.5	Objectives	33
2.2	Methods	35
2.2.1	Connectivity hash codes	35
2.2.2	Candidate retrieval	40
2.2.3	Candidate reasoning and classification	42
2.3	Results and discussion	46
2.3.1	Exploratory analysis of aligning <i>iJR904</i> and MetaCyc 16	46
2.3.2	Hash code collision analysis	49
2.3.3	Structure key distributions	55
2.3.4	Alignment of <i>iJR904</i> and MetaCyc 17.5	59
2.3.5	Alignment of <i>iYO844</i> and <i>iBsu1103</i>	69
2.4	Conclusions	78
2.5	Availability	80
3	Annotation of reconstructions	83
3.1	Introduction	83
3.1.1	Motivation	83
3.1.2	Existing software	84
3.1.3	Objectives	85
3.2	Methods	86
3.2.1	Data representation	86
3.2.2	Data import and export	90
3.2.3	Data resources	95
3.2.4	Graphical interface	99
3.3	Results and discussion	104
3.3.1	Cross-references	104
3.3.2	Chemical structure annotations	108
3.3.3	Verification	112
3.3.4	Standardisation	112
3.4	Conclusions	115
3.5	Availability	117

II	Chemical information processing	119
4	Ring perception	121
4.1	Introduction	121
4.1.1	Motivation	121
4.1.2	Types of ring information	122
4.1.3	Objectives	126
4.2	Methods	129
4.2.1	Graph data structures	129
4.2.2	Cycle membership	131
4.2.3	Minimum cycle basis, essential and relevant cycles	132
4.2.4	All elementary cycles	134
4.3	Results and discussion	135
4.3.1	Cycle membership	135
4.3.2	Minimum cycle basis, essential, and relevant cycles	137
4.3.3	All elementary cycles	141
4.4	Conclusions	143
4.5	Availability	144
5	Aromaticity, Kekulisation, and tautomerism	147
5.1	Introduction	147
5.1.1	Motivation	147
5.1.2	Aromaticity	148
5.1.3	Kekulisation	149
5.1.4	Tautomerism	150
5.1.5	Objectives	151
5.2	Methods	153
5.2.1	Aromaticity	153
5.2.2	Graph matching	154
5.2.3	Fast Kekulisation	158
5.2.4	Unique tautomer generation	160
5.3	Results and discussion	161
5.3.1	Aromaticity	161
5.3.2	Kekulisation	162
5.3.3	Unique tautomer generation	167
5.4	Conclusions	170

5.5	Availability	171
6	Stereochemistry	173
6.1	Introduction	173
6.1.1	Motivation	173
6.1.2	Stereocentres	173
6.1.3	Describing stereochemistry	174
6.1.4	Objectives	175
6.2	Methods	177
6.2.1	Representation	177
6.2.2	Identification	178
6.2.3	Depiction	182
6.2.4	Comparison	185
6.2.5	Description	187
6.3	Results and discussion	190
6.3.1	Identification	190
6.3.2	Depiction	192
6.3.3	Description	192
6.4	Conclusions	197
6.5	Availability	198
7	Conclusions	201
	Appendices	207
A	Chapter 2 supplementary material	209
A.1	InChI API differences	209
A.2	Description of cycle notation	210
A.3	Guide to database identifiers	210
A.4	MetaCyc structure corrections	211
A.5	Formula key implementation	211
A.6	Problematic tetrapyrrole representations	212
A.7	Alignment of <i>iJR904</i> and MetaCyc 17.5	213
A.7.1	Without structure annotations	213
A.7.2	Stereoisomer candidates	213
A.7.3	Structurally related candidates	215

A.7.4	False negatives	216
A.7.5	One candidate	217
A.7.5.1	Identical false positives	217
A.7.5.2	Stereoisomer false positives	218
A.7.5.3	Structurally related false positives	218
A.7.6	Two or more candidates	219
A.7.6.1	Two candidates, false positives	219
A.7.6.2	More than two candidates, false positives	220
A.7.6.3	Two candidates, true positives	221
A.7.6.4	More than two candidates, true positives	222
A.8	Alignment of <i>iYO844</i> and <i>iBsu1103</i>	222
A.8.1	Without structure annotations	222
A.8.2	False negatives	224
A.8.3	One candidate	224
A.8.3.1	Identical false positives	224
A.8.3.2	Stereoisomer false positives	225
A.8.3.3	Structurally related false positives	225
A.8.3.4	Structurally related true positives	226
A.8.4	Two or more candidates	227
A.8.4.1	Two candidates, false positives	227
A.8.4.2	More than two candidates, false positives	228
A.8.4.3	Two candidates, true positives	229
A.8.4.4	More than two candidates, true positives	231
B	Chapter 3 supplementary information	233
B.1	Incorrect MIRIAM annotations	233
B.2	SBML species definitions	234
B.3	Normalisation of metabolite names	235
B.4	Ring closing SMARTS patterns	235
B.5	Convenience functionality	236
C	Chapter 4 supplementary information	239
C.1	Introduction to graph theory	239
C.2	Determining cycle membership with a spanning tree	240
C.3	Drawbacks of using a timeout with AllRingsFinder	241
C.4	Performance, absolute time taken	242

C.4.1	Cycle membership	242
C.4.2	Minimum cycle basis	243
C.4.3	Relevant cycles	243
C.4.4	Essential cycles	244
C.4.5	All elementary cycles	244
C.5	Feasibility of all elementary cycles	245
C.5.1	Number of cycles found	245
C.5.2	Number of infeasible structures	245
D	Chapter 5 supplementary information	247
D.1	Alternative aromaticity algorithms	247
D.2	Backtracking tautomer assignment	247
E	Chapter 6 supplementary information	249
E.1	Verifying stereocentre comparison	249
E.2	Differences in tetrahedral perception	250
E.3	Unambiguous representation of tefluthrin	252
E.4	Differences in Cahn-Ingold-Prelog	252
	References	257

List of Figures

1.1	Major steps in the creation of a metabolic reconstruction	3
1.2	Overview of data resources for metabolic reconstruction	6
1.3	Example formats for chemical representation	14
2.1	Network of direct and indirect cross-references in Recon 2	28
2.2	Example of canonically labelling phenol	32
2.3	Ignoring hydrogens to maintain correct stereochemistry	38
2.4	Bottom-up candidate retrieval	41
2.5	Top-down candidate retrieval	42
2.6	<i>iJR904</i> metabolites matched in MetaCyc	47
2.7	Difference identified between metabolites	48
2.8	Duplicate ChEBI structures identified by the hash code	50
2.10	Structure key performance	56
2.9	Structure key distributions	57
2.11	Matching and classification of <i>iJR904</i> to MetaCyc 17.5	61
2.12	Unspecified stereocentre in <i>iJR904</i>	62
2.13	Unspecified stereocentre in MetaCyc	63
2.14	Classification of reasons a single metabolite matched two entries	64
2.15	Variable attachment to coenzyme A	65
2.16	Duplicate entries in MetaCyc	66
2.17	Tautomers present in BiGG	67
2.19	Different representations of 2-dehydro-3-deoxy-D-gluconate	69
2.18	Matching and classification of <i>iYO844</i> to <i>iBsu1103</i>	70
2.20	Duplicate entries in <i>iBsu1103</i>	72
2.21	Differences in fatty acid structure annotations	73
2.22	Reversed dipeptide representations	75

2.23	γ -attached amino acids	76
2.24	Different structure annotations of crotonobetaine	76
2.25	Duplicated structure of tetradecanoate	77
2.26	Fischer projections result in multiple retrieved candidates	77
2.27	Candidate matches for ornithine	80
3.1	Diagram of object data model and interactions	86
3.2	SBML encoding of metabolite species	87
3.3	Diagram of generic and specific annotation types	90
3.4	Selection of a compatible binary reader	91
3.5	Screenshot of configuring tabular import	92
3.6	Expansion of implicit terms when parsing reaction equations	94
3.7	Architecture for creating a local data index	98
3.8	Annotated screenshot of Metingear interface sections	100
3.9	Screenshot of table display	101
3.10	Screenshot of inspector display	102
3.11	Input suggestion for an organism name	102
3.12	Screenshot of manually entering annotations	103
3.13	Overview of available annotation methods	104
3.14	Adding a cross-reference in the entity table	105
3.15	Screenshot of the extraction of annotations from free text notes	106
3.16	Specifying resource priority for automated cross-referencing	107
3.17	Manual selection of suggested cross-references	108
3.18	Suggested cross-reference quality indication	108
3.19	Screenshot of configuring oligopeptide generation	110
3.20	Generated structure for an fatty acid attached to the ACP	110
3.21	Generated structure for an coenzyme A attached metabolite	111
3.22	Screenshot of selecting a tautomer for guanine	113
3.23	Screenshot of selecting ring-chain tautomers	114
4.1	Short cycles of CHEBI:36471	123
4.2	Minimym cycle basis and systematic naming of barrelene	124
4.3	Elementary cycles of azulene	125
4.4	Elementary cycles of anthracene	126
4.5	Elementary cycles of porphyrin	127
4.6	Data structures for representing graphs	130
4.7	Performance improvement of calculating cycle membership	136

4.8	Performance improvement due to pre-processing	138
4.9	Performance improvement of calculating a minimum cycle basis	138
4.10	Performance improvement of calculating the essential and relevant cycles	139
4.11	Relative performance comparison of unique and non-unique cycle sets .	140
4.12	Performance improvement of calculating all elementary cycles	142
5.1	Kekulé representations of o-xylene	148
5.2	Kekulisation of azulene with CDK algorithms	149
5.3	Interpretation of pyran-4-one by different toolkits	150
5.4	Examples of tautomerism due to a mobile hydrogen	150
5.5	Rearrangement of naphthalene to a bipartite depiction	154
5.6	Multiple perfect matchings of naphthalene and relation to Kekulé forms	155
5.7	Augmenting paths to increase match cardinality	156
5.8	Blossoming odd cycles to find non-shortest augmenting paths	157
5.9	Example of greedy matchings for pyrrole	159
5.10	Difference in aromaticity perception between CDK 1.4 and 1.5	161
5.11	Invalid input due to tetravalent carbon anion	163
5.12	Invalid input due to a radical	164
5.13	Preferred Kekulé forms of biphenylene	166
5.14	Equivalent Kekulé forms of 1-methylpentalene	167
5.15	A tautomer not considered by the InChI	168
5.16	Neutralisation normalisation performed by the InChI	169
6.1	Data structures for storing stereochemistry	177
6.2	Determining configurations of geometric isomers	179
6.3	Wedge and hatch bond conventions	180
6.4	Ambiguous stereocentre in PubChem-Compound	180
6.5	Reverse hatch depiction in MetaCyc	180
6.6	Partial stereochemistry encoded by the InChI	181
6.7	Depicting tetrahedral centres (four atoms)	182
6.8	Depicting tetrahedral centres (three atoms)	183
6.9	Depicting tetrahedral centres (three atoms) in rings	184
6.10	Example of extended tetrahedral depiction	184
6.11	Different absolute stereochemistry labels in a substructure	185
6.12	Comparing stereochemistry data structures	186
6.13	Examples of symmetric stereochemistry	188
6.14	Using auxiliary descriptors to label intradependent stereocentres	189

6.15	Problematic tetrahedral stereocentres	190
6.16	Corrupting stereochemistry depiction	192
6.17	Variable CIP descriptors of myo-inositol using JChem	193
6.18	Labelled stereocentres that were ignored by JChem	195
A.1	InChI API differences	209
A.2	Cycle notation	210
A.4	Tetrapyrrole representations	212
B.1	Screenshot of splitting a metabolite	236
B.2	Screenshots of sequence homology	237
C.1	Effect of timeout on coverage	241
C.2	Effect of timeout on performance	242
D.1	Combining aromatic rings	247
D.2	Backtracking tautomer generation	248

Glossary

API	Application Programming Interface
AtomContainer	<i>chemical structure data type of the CDK library</i>
CDK	Chemistry Development Kit
ChEBI	Chemical Entities of Biological Interest
CIP	Cahn-Ingold-Prelog
EC	Enzyme Commission
GEM	Genome-scale metabolic model
GPR	Gene-Protein-Reaction
InChI	IUPAC International Chemical Identifier
KEGG	Kyoto Encyclopedia of Genes and Genomes
MIRIAM	Minimum Information Required In the Annotation of Models
MDK	Model Development Kit
PRNG	Pseudo-Random Number Generator
RDF	Resource Description Framework
SBML	Systems Biology Markup Language
SMARTS	SMiles ARbitrary Target Specification
SMILES	Simplified Molecular-Input Line-Entry System
$O(\dots)$	<i>asymptotic upper bound of a computation as function of the input size</i>
XML	Extensible Markup Language
XOR	disjoint union (exclusive or)

Chapter 1

Introduction

Characterising complex biological processes requires both a component and system level view. Systems biology seeks to describe and reason about the diverse network of interactions inside cells. In turn, the systems view can generate hypothesis and discover emergent properties that can be tested and verified at the component level. The functions of a cell can be organised and studied as a cellular wiring diagram, a genetic circuit. Different realisations of genetic circuits have been created and applied to multiple cellular processes, including but not limited to: regulation, signalling, and metabolism.

The genetic circuit for metabolism describes chemical transformations of small molecules catalysed by enzymes within a cell. Assembly of the circuit as an enzymatic reaction network allows the study of the metabolic routes (pathways) and prediction of observable characteristics, the phenotype. Genome-scale metabolic models (GEMs) have been constructed and experimentally verified for more than 78 species^[1,2]. Among other uses^[3], they have been used to predict lethal gene deletions, study bacterial evolution, and discover biological function^[4]. The most exciting application is in the optimisation or modification of metabolism^[5], in some cases for the production of *de novo* metabolites^[6].

1.1 Introduction to genome-scale metabolic models

GEMs are constructed from genome and literature information. The assembly of this information provides a network of reactions that an organism is capable of performing. The mathematical formulation of the network as a model enables the simulation, measurement, and prediction of metabolic states. The term *reconstruction* is used in this thesis in reference to the network or model

1.1.1 Constraint-based analysis

The dynamic properties of metabolic reaction networks can be accurately studied through kinetic modelling. These models provide the concentration of metabolites in time but require extensive parametrisation and reaction specific rate laws. A rate law describes how the reaction rate changes at different substrate concentrations. Difficulties in measuring accurate rate laws hinder the precise modelling of large-scale networks but this can be accomplished through generic rate laws and data integration^[7,8].

At certain time scales, metabolism can be viewed as a steady state. Under this assumption all metabolites are consumed and produced at the same rate and precise concentrations are not determined. Instead, the flow of metabolites through the network is analysed through the distribution of reaction fluxes. Only the quantities of reaction substrates and products (stoichiometries) are required. The little amount of parametrisation allows the simulation of much larger genome-scale networks.

There may be multiple steady states that a network is capable of. In reality, pressures from the environment and physical limitations mean some of these states are infeasible. By applying constraints to the reaction fluxes the set of possible states can be restricted. Constraints are added by bounding a reaction flux with upper and lower limits. This constraint-based analysis provides an effective framework to model different properties and integrate experimental observations^[9].

Flux balance analysis (FBA) uses the steady state assumption and an objective function to model a desired phenotype as an optimisation problem^[10]. To model the growth of an organism, the objective function can be to maximise the rate of a *biomass* reaction. The biomass reaction defines relative stoichiometric coefficients of metabolic precursors that a cell requires for replication and maintenance. By deleting genes and

constraining nutrient uptake, the relative effect on growth can be observed through changes in the biomass reaction flux.

1.1.2 Creation

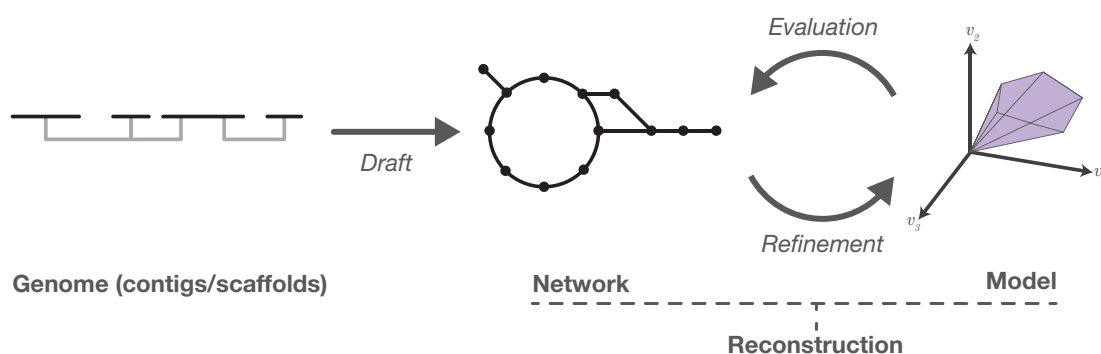


Figure 1.1: Genome-scale reconstructions are created from genome sequence information. The annotation of the sequence with enzymatic function provides a draft network of biochemical reactions. Refining the network and converting it to a model allows simulation. The simulation can be used to evaluate missing or erroneous properties and therefore improve the network.

The creation of metabolic reconstructions can be organised into three steps: creation of a draft, refinement, and evaluation (Fig. 1.1). A detailed protocol for generating high-quality reconstruction is described by Theile I and Palsson B^[11], the main steps of which are summarised here.

Creating a draft

The creation of a draft reconstruction starts with the genome sequence. The fully assembled genome sequence may not be available but sequence contigs and scaffolds are sufficient. Open reading frames and gene locations in the sequence are predicted^[12,13]. These genes are then assigned a putative function using sequence homology; functional roles of parts of the sequence may also be determined^[14].

The predicted functions provide a complete parts list of the cells known capabilities. For metabolism, the functional annotations of most interest are those of enzyme and transporter coding genes. Enzyme functions are usually described indirectly through the Enzyme Commission (EC) number. The EC number specifies a hierarchical classification of biochemical functions^[15]. Promiscuity of enzymes means it is possible

that a single protein may have multiple EC numbers assigned as they catalyse more than one reaction^[16]. Analogous to the EC numbers are Transport Classification (TC) numbers^[17] for transporters.

These functional annotations infer biochemical transformations and movements of metabolites into and out of the cell. By listing and connecting the metabolites involved in the transformations, a network can be formed. The network now describes not only the biochemical capabilities but also the interaction between those capabilities.

Refinement

The network generated from functional annotation may be incomplete. The refinement stage seeks to correct errors and include missing functionality. It is closely coupled to evaluation, as the reconstruction is iteratively verified. The refinement stage of high quality reconstructions require meticulous manual curation. Model reconstruction and refinement jamborees have been organised for some organisms^[18].

The functional annotation of genes is limited by pre-existing (and organised) knowledge and the accuracy of prediction. Ideally, each reaction should be supported with experimental data or literature evidence on the organism or related species. Literature may also guide the inclusion of reactions that were not predicted by sequence homology, are spontaneous, or carried out by enzymes of unknown sequence^[19].

Depending on the source of reactions, several adjustments may be required. Reaction stoichiometries are balanced and metabolite charge adjusted to physiological pH. The reversibility (direction) of the reaction should also be assigned. Multi-compartment reconstructions require defined reaction locations and intracellular transporters. Localisation of reactions can be inferred through the prediction of the enzyme location^[20]. Reactions may be carried out by multiple subunit enzymes, complexes, or isozymes. The Gene-Protein-Reaction (GPR) associations characterise these relationships as logical statements and are required for evaluating gene deletions.

Annotation of the components (reaction, metabolites, enzymes, etc.) in the reconstruction with pathways, subsystems, citations, and external database identifiers facilitates their interpretation and reuse. Metabolites may also be identified using chemical structures, that serve as database independent identifiers.

Evaluation and simulation

The evaluation stage identifies problems with the reconstruction and feeds these back for correction in the refinement stage. Problems with the reconstruction are identified either by inspection of network components (i.e. mass balance) or through simulation.

The conversion of the network to a simulation requires the inclusion of several additional components. The components include the biomass, exchange, sink and demand reactions. Although they form part of the network, they are primarily to allow simulation and specific to each reconstruction. The role of biomass and exchange reactions was previously introduced in the context of FBA. Demand and sink reactions are included to drain metabolites to an intracellular pool^[11]. The biomass reaction will usually encode unspecific or “lumped” metabolites. These represent average compositions are specific to each reconstruction.

Dead-ends are metabolites that are present but cannot be produced (or consumed) and are the result of missing or blocked reactions. These metabolites constitute gaps, either in knowledge or actually in the network. The networks gaps can be resolved (unblocked) by reversing or adding reactions^[21].

Gene knockouts can be simulated and verified experimentally. The model can then be refined such that *in silico* growth / no growth deletions correspond to *in vivo* observations^[22]. The model can also be checked as to whether known metabolites are produced on a given growth medium and if the growth rate is faster or slower than expected.

1.1.3 Exchange

Traditionally the de facto standard for distributing reconstructions was as tables (Microsoft Excel spreadsheets) listing reactions, metabolites and gene products. Differences in the structuring of the tables restricted their portability. More common now is the use of open interchange formats encoded as extensible markup language (XML). Two formats for representing models are CellML^[23] and the SBML (System Biology Markup Language)^[24]. The BioPax^[25] standard is also notable but is mainly used for the exchange of pathway information.

Standard software libraries for interpreting the formats are valuable in storing and conveying model information. The popularity of SBML has grown rapidly and now

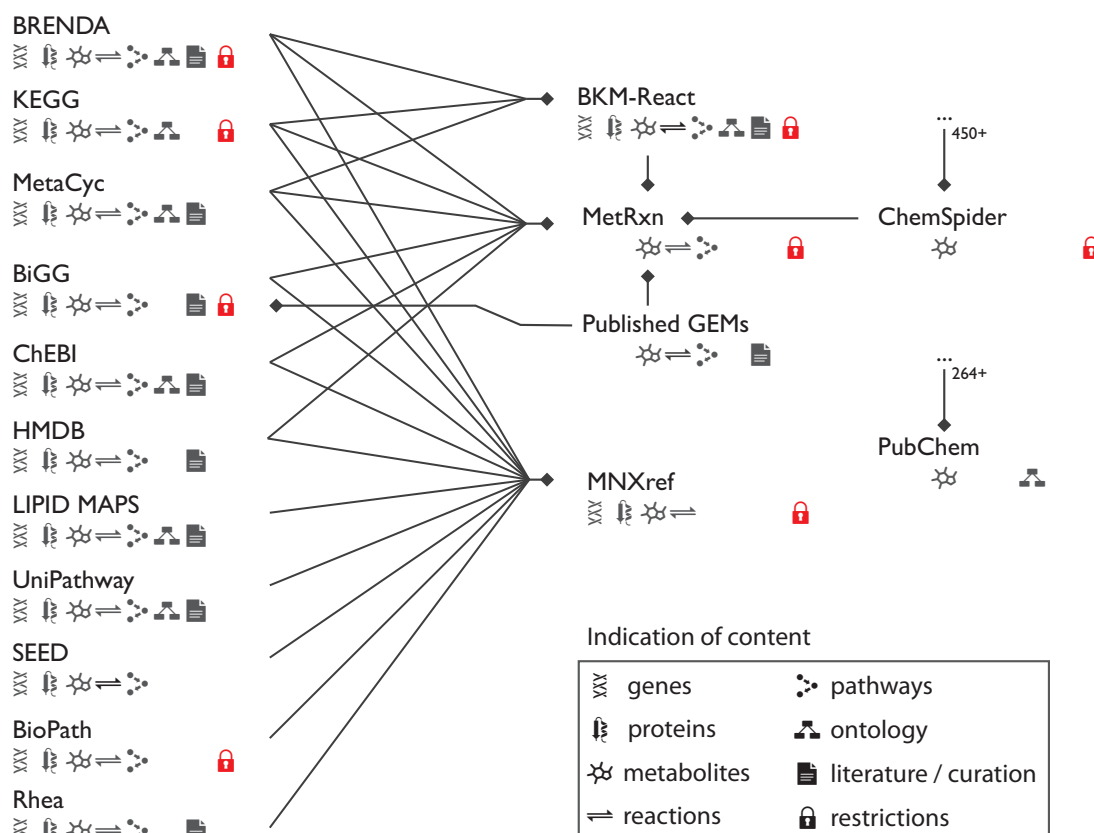


Figure 1.2: Major data resources: composition and contents. Presence of an icon indicates information of that type is available. This information may be directly contained in the resource or indirectly linked to another. A red lock indicates restrictions apply to the data (only free for non-commercial use) or that the resource is not publicly distributed (available on request).

over 250^[26] tools have at least some support for reading or writing the format. Standard formats also encourage the use of resolvable database identifiers for improved annotation and reuse^[27].

Although tabular formats are still in use, reconstructions are now commonly distributed as SBML and deposited in the BioModels^[28] repository.

1.1.4 Databases

For creating reconstructions, metabolic information is obtained from multiple data sources including: databases, literature review, and expert knowledge. The boundaries between the content of databases is fading as more resources integrate related in-

formation. In general, the content in databases can be considered layered, such that reaction databases will also include information on metabolites and pathway databases will also include information on reactions. This information may be included in the database directly or indirectly via cross-reference (Fig. 1.2).

The BiGG database is a repository of 10 experimentally verified models (Apr 2014)^[29]. Although small, the high-quality reconstructions serve as an excellent source of information. Data can be download as SBML but metabolites, genes, and proteins are only identified by name.

The Kyoto Encyclopaedia of Genes and Genomes (KEGG)^[30,31] is a popular metabolic resource. The database is divided into several intradependent collections. Of most interest for metabolic reconstruction is the KEGG LIGAND resource, that comprises three sub databases with information on: compounds, glycans, and reactions. Reaction information is in turn split into: reactions, pair alignments (RPAIR), reaction classes, and enzymes. Since December 2011, the KEGG resource requires a paid subscription for bulk access.

BioCyc is a collection of organism-specific Pathway / Genome Databases (PGDBs) or Cycs (for encyclopaedia)^[32,33]. PGDBs are a rich source of information on pathways, reactions, metabolites, genes, and gene products. There are three tiers of PGDBs, lower levels have more manual curation and are available for model organisms (e.g. EcoCyc for *Escherichia coli*). The MetaCyc database serves as the main repository of data from which the organism specific PGDBs are derived^[34].

Other pathway databases are: Reactome^[35], UniPathway^[36], and WikiPathways^[37]. Reactome and UniPathway approach pathways from genome annotation but biochemical information is included. WikiPathways is an exciting initiative in crowdsourcing information on signalling and metabolic pathways.

Enzyme specific databases include ENZYME^[38] and IntEnz^[39]. These resources index biochemical reactions and are indirectly linked to sequence information via cross-reference. BRENDA^[40,41] provides a comprehensive repository of enzymatic knowledge but is difficult to obtain data from; limited downloads are only provided as a text description. Rhea^[42] is a curated resource for reaction information linked to ChEBI. Unlike reactions in reconstructions, enzyme and reaction databases do not encode a specific compartment, except for a generic *in* or *out* annotation for transport reactions. The TCDB and TransportDB^[43] provide transporter specific protein information; reaction equations with individual metabolites are not provided.

Information specific to metabolites can be obtained from chemical databases. There are a large number of databases that vary in scope and may contain non-metabolic (inorganic) structures. Chemical Entities of Biological Interest (ChEBI) provides a manually curated collection of small molecules and chemical ontology^[44]. The ontology describes relationships between entries and categorises them in a hierarchical manner. The Human Metabolome Database (HMDB) contains metabolites and spectral measurements specific to human metabolism as well as tissue and cell type specific concentration ranges^[45]. Lipids are an extensive set of biological compounds with several dedicated resources including the LIPID MAPs structural database (LMSD)^[46] and LipidHome^[47].

It is often the case that information from more than one of these databases is required. Cross-references between resources are provided by databases internally or externally via identifier mapping^[48].

Alternatively, a merged database can be created that combines entries from multiple child databases into a single unified (parent) resource. The BioWarehouse^[49] framework provides a database scheme and tools that enables the creation of a reconciled metabolic resource. BKM-React^[50] provides integrated access to the reactions from BRENDA, KEGG and MetaCyc databases. MetRxn^[51] integrates eight different databases and 90 GEMs (Mar 2014). The MNXref^[52] is an impressive metabolic resource that reconciles many child databases and provides high-quality downloads.

For chemical information the PubChem (Compound and Substance)^[53] and ChemSpider^[54] resources integrate several hundred resources.

1.1.5 Software tools

The creation of metabolic networks and their conversion to models is a complex process. The integration of many methods are required in their creation. For brevity, tools for the functional annotation of sequences are not discussed.

SimPheny (Simulated Phenotypes)

SimPheny is a platform for reconstruction, annotation, and simulation^[55]. The software is proprietary and originally developed by Genomatica Inc.; little information is publicly available.

Pathway Tools, PathoLogic, and MetaFlux

Pathway Tools is a collection of software for the construction, maintenance, and analysis of a PGDB^[56]. PGDBs created with the software are distributed as the BioCyc collection of databases. The PathoLogic software uses the genome annotation and MetaCyc to predict metabolic pathways for an organism. Missing reactions (gaps) in pathways are identified and filled by comparison and scoring against existing pathways^[57]. The pathways can be assembled into a connected network and model; constraint-based analysis is available using MetaFlux^[58]. Gaps in the reconstruction are resolved by an optimisation method.

model-SEED and RAST

The model-SEED web application is an automated pipeline for high-throughput creation of draft reconstructions^[59]. Although described as draft, steps listed in the refinement and validation stages are included. The auto-complete optimisation procedure resolves dead-end metabolites using a score based system and produces a simulation ready model. If cell phenotype data is available, validation of gene deletions is performed.

Sequences are first annotated with function and subsystems using the RAST service^[60]. The model-SEED primarily utilises information from an in-house database (SEED) composed of KEGG of models.

The SuBliMinaL toolbox

The SuBliMinaL toolbox is a collection of tools for automating the creation of a model from existing networks^[61]. Reconstructions from SBML, KEGG, and BioCyc can be merged, creating a consensus. Annotations are imported or added with libAnnoationSBML^[62]. Prior to merging reconstructions, reactions are balanced and metabolites protonated to physiological pH. Metabolites are merged using the ChEBI ontology, this allows a relaxed identity check by inspected relationships. Compartmentalisation, transporter addition, and biomass definition are also performed. The toolbox has been used in the creation of consensus reconstructions^[63,64] and in Path2Models^[65], for the automated generation of a large collection of constrained-based models.

RAVEN (reconstruction, analysis and visualization of metabolic networks)

RAVEN is a MATLAB library (and associated binaries) for creating and analysing reconstructions from genome sequences^[66]. New reconstructions are created through inspecting homology with existing published models and KEGG. Metabolite entries are merged based on primary identifier or name. Existing models are used as the primary data resource, as they provided a good source of reaction directionality and compartmentalisation. Reactions are automatically compartmentalised using an approach based on both predicted location and network topology.

MetaNetX

The MetaNetX web application is a cloud based workspace for the modification, comparison, and analysis of existing models^[67]. The repository data is stored in the MNXref database. Models can be deposited, merged, and analysed. Constraint-based analysis and refinement can be performed; solutions to unblocked reactions are scored using network motifs^[68].

COBRA (constraint-based reconstruction and analysis)

The COBRA toolbox^[69,70] and COBRApy^[15] are MATLAB and Python toolkits for constraint-based modelling. The toolkits provide many methods required when evaluating and refining reconstructions. An extension to the COBRA library, rBioNet, provides a graphical interface and database to create metabolite and reaction entries^[71].

Other tools

IdentiCS^[72] identifies enzyme coding sequence and assembles a metabolic network from a genome using KEGG. The GEM System^[73] uses an improved and balanced reaction set, gaps are detected and filled by inspecting known pathways. MetNet-Maker^[74] is a desktop application for creating and manipulating KEGG reactions and pathways in SBML format. The Merlin^[75] desktop application provides genome to network creation primarily based on KEGG annotations. A notable feature of Merlin is the handling of transporters via a reconciled TCDB.

GEMSiRV provides reconstruction editing and simulation^[76]. Draft reconstructions are built with the MrBAC^[77] web server. FAME^[78] is a web application that can simulate, merge, or create a model from a list of KEGG Pathways. MicrobesFlux^[79] is a cloud based web application that provides simulation and creation of models by selecting KEGG reactions. The MEMOSys database provides a framework for storage and management of reconstructions, including version control^[80]. OptFlux^[81] is a desktop application for performing constraint-based analysis and strain optimisation.

1.2 Introduction to cheminformatics

Although reconstructions can include the modelling and integration of macromolecular processes^[82], the majority focus is on biochemical reactions of small molecules. The chemistry of these small molecules may be described in varied levels of detail. The majority of reconstructions include common or trivial names to describe metabolites, as well as the molecular formula and charge. A cross-reference to an external database identifier allows additional information to be retrieved. The structural formula of a metabolite provides an unambiguous description of what the metabolite is. Uses for the chemical structure in the creation and analysis of reconstructions will be introduced shortly.

Cheminformatics (or chemoinformatics) deals with the computational storage, retrieval, and reasoning about chemical information. The basic concepts and uses are summarised here; for a comprehensive introduction please refer to the Handbook of Chemoinformatics^[83] and the Handbook of Chemoinformatics Algorithms^[84].

1.2.1 Representation and applications

Graph theory

Chemical structures can be effectively modelled as a labelled graph^[85]. More descriptive models have been proposed^[86,87] but the simplicity and efficiency makes it applicable to handling datasets with millions of entries.

A *graph* consists of a set of *vertices* (nodes) and *edges* (connections). An *edge* is a pairing of two *vertices* known as the *end points*. Two *vertices* are *adjacent* if there exists an *edge* for which the two *vertices* are *end points*. Each *end point* is *incident* to the *edge*. A graph is *labelled* by associating values with the *vertices* and *edges*, in a chemical *graph* each atom is associated with a *vertex* and each bond with an *edge*. In a Lewis structure, atoms are described by the chemical element and the charge whilst the bonds are described as the number of bonded electron pairs (bond order). In practice additional properties and labels are also stored on the atoms and bonds.

Identity

As a graph, two structures are equivalent if there is a *bijection* (one-to-one mapping) between the *vertices* that preserves the *adjacency* relation. The two graphs are then said to be *isomorphic*. Similarly, a substructure can be described by finding a *subgraph isomorphism* where the vertices and edges of one structure are a subset of another.

Structure queries provide a powerful tool for reasoning about structures, here atoms and bonds may be a logical predicates that match multiple discrete elements or properties.

Similarity

The concept of similarity can be computed in several ways. The maximum common (edge) subgraph (MCS/MCES) can be used to identify a common substructure between two compounds^[88,89]. A similarity score can be calculated by inspecting the intersection and complement of the shared vertices and edges.

Unfortunately the computational complexity of MCS makes it intractable for even moderately sized structures. The comparison of structural features (or keys) can be used instead of the full connected graph. Through only counting matched and unmatched structural features a more efficient, but estimated, similarity can be calculated. Features are essentially substructures, they may be predefined or enumerated. By hashing features, a structure can be summarised in a fixed size (usually binary) vector known as a fingerprint. Fingerprints allow very rapid comparison and searching^[90].

A less common measure of similarity is the graph edit distance; this measure describes the number of modifications (edits) required to transform one graph into another^[91].

Descriptors

Similarity is of interest primarily because similar compounds have similar properties and thus biological effect^[92]. The fingerprints discussed in the previous section describe the connectivity of a compound but other attributes can also contribute.

The term *Descriptor* encompasses a broad range of values that may be topographical, geometrical, physicochemical, or mathematical. Computing descriptors can be used

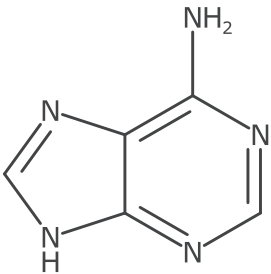
	Name	adenine
	Systematic Name	9H-purin-6-amine
	SMILES	<chem>NC1=C2N=CNC2=NC=N1</chem> <chem>Nc1ncnc2[nH]cnc12</chem> <chem>NC=1N=CN=C2NC=NC12</chem>
	InChIKey	GFFGJBXGBJISGV-UHFFFAOYSA-N
InChI InChI=1S/C5H5N5/c6-4-3-5(9-1-7-3)10-2-8-4/h1-2H,(H3,6,7,8,9,10)		
molfile (CTab V2000) <pre> Vdr0541 06091411492D 10 11 0 0 0 0 999 V2000 3.2789 2.0625 0.0000 N 0 0 0 0 0 0 0 0 0 0 0 0 3.2789 1.2375 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 ... 1.7798 1.0799 0.0000 N 0 0 0 0 0 0 0 0 0 0 0 0 2.5645 0.8250 0.0000 C 0 0 0 0 0 0 0 0 0 0 0 0 1 2 1 0 0 0 0 2 3 1 0 0 0 0 ... 2 10 2 0 0 0 0 6 10 1 0 0 0 0 M END </pre>		

Figure 1.3: Examples of formats representing adenine. Multiple SMILES are listed to emphasise there are multiple valid representations, the atom and bond blocks of the molfile example are truncated.

in the construction of quantitative structure-activity relationship (QSAR) models for the prediction of emergent chemical properties.

1.2.2 Exchange

There are multiple formats for the exchange and distribution of chemical data^[93] (Fig. 1.3). Each format has advantages that may be specific to an application domain or software vendor. In the context of exchanging metabolic information several formats have arisen as de facto standards. In addition to those discussed here, the Chemical Markup Language (CML) is worth noting as a community driven XML standard^[94]. CML can be customised and extended through the use of conventions.

Naming and nomenclature

Chemical structures may be referred to by name. Trivial or common names serve as a reference to a structure but can be ambiguous. Systematic names describe the structure

using formal set of rules that can be interpreted^[95] and generated^[96] computationally.

molfile

The molfile is a member of the CTfile (Connection Table) family of formats originally developed by MDL information systems and now maintained by BIOVIA^[97]. The other formats in the family store reactions (rxnfile), generic queries (RGfile), and collections (SDfile and RDfile).

CTfiles are a depiction based format that store atom locations as two- or three- dimensional coordinates. Atoms and bonds are listed in a fixed width format. More advanced features include queries, generic structures, and Sgroups (substructure groups). The generic structures allow description and combination of structure attachments (R groups) and are primarily useful for queries. Sgroups allow the representation of higher level concepts including: polymers, mixtures, and display abbreviations^[98].

SMILES

SMILES (Simplified molecular-input line-entry system) is a concise line-notation that was first created at the US Environmental Protection Agency^[99] and further developed by Daylight Chemical Information Systems^[100]. The SMILES format was originally designed for entry into computing terminals and comprises a set of simple rules allowing for efficient interpretation and generation by a computer or user^[101].

Since only connectivity information is stored and not coordinates, the format has very high information density compared to depiction formats like the molfile. Coordinates and additional atom information can be stored as an auxiliary data vector and included if required.

Formally, a SMILES string is a spanning tree created by a depth-first traversal of the a chemical structure. By uniquely ordering the atoms and bonds, a canonical SMILES string can be encoded^[102]. The SMILES can then be used for fast retrieval from databases and identity checks. It should be noted that a SMILES string does not need to be canonical for simply representing and storing a structure.

The simplicity and popularity of SMILES has led to different software implementations. Small differences in the algorithm used to order the atoms means canonical

SMILES should only be compared if they were created using the same software tool and version.

Related to SMILES is the SMARTS (SMiles ARbitrary Target Specification) format that allows structural queries to be expressed in a compact linear notation^[103].

InChI

The IUPAC International Chemical Identifier (InChI) was developed by the National Institute of Standards (NIST) and the International Union of Pure and Applied Chemistry (IUPAC)^[104]. The InChI is a unique identifier for structural information. The InChI toolkit is a command line program to convert molfile and CML input into InChIs. The toolkit can produce different identifiers based on input options; the standard InChI provides an identifier that can be directly compared between resources. A key difference from SMILES is that the single implementation means standard InChIs are comparable.

The InChIKey is a fixed size hashed digest of the full identifier. As a hashed representation, collisions are rare but possible^[105,106].

InChI development is continuing and future versions will include generic structures (Rgroups) and polymers^[107]. It is not yet known how newer releases will be compatible with the current standard InChI.

1.2.3 Software libraries

There are multiple software libraries available that assist in the manipulation of chemical information. Libraries primarily vary in the programming language in which they are written, features available, performance, and license. The features provided by the libraries are complementary, with some aimed at specific goals such as format conversion (Open Babel) or machine learning (RDKit). Although commercial libraries are often available for free to academia, creating and releasing derived software or services is restricted. Restrictions can even cover the redistribution and use of calculated properties.

Some commonly used toolkits are:

Name	Source	Main language	Available
CACTVS ^[108]	Closed	C, Tcl	http://www.xemistry.com/
CDK ^[109,110]	Open	Java	http://sourceforge.net/projects/cdk/
Daylight	Closed	C	http://www.daylight.com/products/toolkit.html
Indigo	Open	C++	http://www.ggasoftware.com/opensource/indigo
JChem	Closed	Java	http://www.chemaxon.com/download/jchem-suite/
OEChem	Closed	C++	http://www.eyesopen.com/oechem-tk
Open Babel ^[111]	Open	C++	http://openbabel.org/
RDKit	Open	C++, Python	http://www.rdkit.org/

Free and open source cheminformatics software libraries^[112] are much less restrictive and encourage their use in creating and releasing derived tools. The Chemistry Development Kit (CDK) is an open source Java library first released in 2001. The library provides features to load, store, represent, and reason chemical information. In addition to Java, the library can also be used in other Java based languages (Groovy, Scala, Clojure), R^[113], Python (via Jython), and Lisp^[114]. The functionality has been integrated in the workflow tools Taverna^[115,116] and Knime^[117].

1.3 Chemical information in metabolic reconstructions

Chemical information can be applied to the creation of reconstructions and exploration of their biochemistry. This section introduces and explores the ways in which chemical structures can be utilised.

1.3.1 Applications

Identification

In a reconstruction, metabolites may only be identified by name or sometimes cryptic abbreviations^[58]. A more robust form of identification is to include cross-references to an external database. These indirect links may be included as a non-standard text pattern or a resolvable identifier. The MIRIAM registry provides a catalogue of data collections allowing resolvable identification of entries in a particular database^[27]. This allows for unambiguous distinction between entries when different database identifiers have the same or similar semantics (e.g. PubChem-Compound and PubChem-Substance).

Since a metabolite may be missing from a database, multiple or reconciled resources may need to be referenced. As reconstructions can include literature information, a metabolite may not be found in any reference database. Some databases allow submission of new entries, but their inclusion may be restricted by the evidence available or the scope of the database. A database may not include an exact match and so a reference may be added to a database entry with different protonation state or partial stereochemistry. An alternative to using cross-reference is to identify metabolites by their chemical structure. The chemical structure is database independent and can precisely represent the metabolite.

The chemical structure of metabolites has been used extensively in the creation of reconciled databases and identifier mappings. The BKM, MetRxn, and MNXref merged metabolic resources all use the chemical structure as the primary source of identifying equivalent entries.

Physiological pH

In constraint-based analysis, the mass and charge of reactions is a fundamental constraint that must be conserved. The information can be stored and derived from a structure representation but is often stored as separate formula and charge annotations. Depending on the pH of a solution, a metabolite may be protonated or deprotonated. It is therefore useful to compute the protonation state at a physiological pH for the each compartment^[118]. Using the chemical structure, the dissociation constants (pK_a) can be predicted and the structure (de)protonated to the specified pH.

Reaction reversibility

The direction and reversibility of reactions can be used to constrain feasible metabolic states^[119] and resolve blocked reactions^[21]. This information may be included from literature and existing reconstructions. When this information is not available, the Gibbs free energy of formation can be estimated using contributions from functional groups^[120,121]. When the energy of formation is estimated as close to zero the reaction may be reversible.

Atom-atom mapping (AAM)

An atom-atom mapping (AAM) is a bijection between the atoms of reactant and product structures in a reaction. AAMs can be created manually, computationally predicted^[122–124], or retrieved from a database (KEGG RPAIR or ARM^[125]). Predictions of AAM are reasonably successful but manual specification may be necessary for some reactions, particularly when multi-steps are condensed to a single transformation. Rahman A *et al*^[122] report a 99% accuracy when compared to a selection of enzymatic reactions in the semi-curated RPAIR databases. AAMs are a prerequisite for several additional applications listed below.

Metabolic Flux Analysis (MFA)

Intracellular fluxes can be estimated using ^{13}C -MFA, a labelled (^{13}C) substrate is introduced and the distribution of labelled metabolic products experimentally measured, for example with mass spectrometry. The measured distribution of fluxes can then be

correlated to the reactions in a network using AAM^[126]. Models incorporating isotope labelling are usually small but a genome-scale mapping has been created for the *iAF1260* model of *Escherichia coli*^[127].

Pathway routing and prediction

AAMs are one method used in tracing metabolite routes in metabolic networks. A metabolic route is a sequence of reactions from a source to a target metabolite. By combining the AAMs of reactions in a metabolic network, the shortest path of source atoms can be traced to the atoms of a desired target^[128]. ReTrace^[129] inspects the AAMs to find plausible pathways by maximising the number of atoms transferred; this avoids the need to define side metabolites. Alternatively the AAMs can be used to define a cost on reactions. MetaRoute^[130] finds shortest routes in from a source to target vertex that preserve at least one carbon atom. RouteSearch^[131] finds pathways that maximises the number of atoms transferred from the source to target metabolites. ReTrace and MetaRoute are based on information from the KEGG database whilst RouteSearch is based on BioCyc and is integrated in Pathway Tools.

Putative reactions and pathways

The bijection described by the AAM can be used to define the chemical transformation between the reactants and products. If two atoms are connected in the reactant but not in the product, a bond has been broken. If they are connected in the product but not the reactant, a bond was formed.

The concept can be generalised as a Bond-Electron Matrix (BEM)^[132] but it is simpler, and more computationally efficient, to think of the transformation as a list of modifications (edits). By applying the modifications, a chemical structure can be transformed into another and the reaction is run *in silico*.

In cheminformatics, a common format for specifying reaction transformations and queries is the SMIRKS^[133] notation. This notation is a mixture of the previously discussed SMILES and SMARTS linear notations and can encode transformations involving stereochemistry and fragments.

Using chemical transformation routes to generate reactions is well established and utilised in designing organic synthesis (retrosynthesis)^[134]. Several methods exploit

these biochemical transformations, to explore enzyme promiscuity and predict novel metabolic routes. The BNICE framework generalised the set of transformations performed by the EC classes^[135]. Biosynthetic pathways are then enumerated and analysed for feasibility; unfortunately the tool is not available for use. The KEGG PathPred web application is similar to BNICE and provides the prediction of pathways based on structural query patterns and RPAIR data^[136]. RetroPath uses chemical structure signatures to generate and rank synthetic routes^[137]. Reaction transformations can also be used in the enumeration of generic reactions for inferring species specific metabolomes^[138].

Similarity and comparison

Similarity between metabolites and reactions can be characterised using the similarity (e.g. fingerprints) in the chemical structures or transformations. The reaction similarity can be used to guide hypothetical reactions or to predict the EC assignment^[139] and cluster enzyme functions^[122].

In addition to using the AAM to predict routes, it is possible to efficiently identify routes using just the structural similarity. The Pathway Hunter Tool (PHT) uses the similarity to identify shortest paths between metabolites^[140]. A similar method uses linear programming to estimate distances and speed up pathway prediction^[141]. The similarity methods in both these approaches are based on the comparison of structural fingerprints.

Metabolite similarity has also been used to identify potential chemical transformations to fill gaps in reconstructions^[142].

1.3.2 Availability of reconstructions with chemical structures

Very few genome-scale reconstructions have included chemical structure annotations at publication. The *iBsu1103* model of *Bacillus subtilis*^[143] includes molfile representations that were used to compute the Gibbs energy and constrain reaction flux. The consensus reconstructions of *Saccharomyces cerevisiae*^[63] and *Homo sapiens*^[64] include SMILES and InChI representations to identify entries. Although metabolite structures were added in *iAF1260* after initial publication, only the isotope mapping and not the structural formula have been made available.

The MetRxn and MNXref databases include reconstructions that have been reconciled with multiple chemical data resources allowing retrieval of chemical structures through annotation transfer. MetRxn includes more than 90 reconstructions but unfortunately only the metabolite names are openly obtainable. MNXref does provide full metabolite details (inc. structures) and includes published models from BIGG, SEED, and the literature.

1.4 Objectives of this thesis

The chemical structures of known metabolites are readily available from metabolic and chemical databases. The introduced applications of the chemical structure in the context of metabolic reconstructions have mainly focussed on single data sources, primarily KEGG. Being able to apply these techniques more easily to organism or cell specific metabolic reconstructions would be beneficial in their creation, analysis, and integration. As has been noted for metabolic databases, most – *“treat chemical structures more as illustrations than as a datafield in its own right”*^[144], this is also true of reconstructions.

The chemical structure is particularly useful as a means of unambiguously identifying a metabolite. The structure is database independent and can directly identify equivalent metabolites without the need for a common reference.

The creation of new and improved reconstructions through merging existing reconstructions is utilised by both The SuBliMinaL Toolbox and RAVEN. This merging is important for larger-scale composite reconstructions and for including information that can not be obtained from the genome sequence. To accomplish this, methods are required to align the contents of reconstructions.

Existing techniques have focussed on using either resource identifiers, absolute chemical representations (e.g. SMILES string), or precomputed relationships (ontologies). Absolute chemical representations have primarily be utilised in creating large merged databases, that need to be created ahead of time. Interpreting the chemical structure at an atomic level can provide an alignment more sensitive to differences in metabolites.

On estimating Gibbs energy of reactions, Haraldsdóttir H^[145] recently noted that - *“the most time consuming step in applying estimation methods ... was the collection of necessary data, such as metabolite structure”*. Semantic cross-references, if available, provide an indirect means for retrieval of chemical structures but are restrictive if a resource is not accessible or multiple different resources are referenced. A better approach is to annotate reconstructions with structural representations directly. Automated and manual curation methods are needed to include precise chemical structure representations within metabolic reconstructions.

Cheminformatics techniques are required to interpret and reason about the chemical data in reconstructions. Open source cheminformatics libraries provide this but are

often lacking in comprehensiveness, robustness, and efficiency.

Towards these goals, this thesis makes the following contributions:

- Rapid and flexible alignment of identical or similar metabolites in reconstructions through chemical structure. A technique based on connectivity hash codes and graph isomorphisms is used as a means of reasoning about the attributes of metabolites and identify matches (Publication 3, in preparation).
- An intuitive desktop application (Metingear) is created for the construction, management, modification, and annotation of reconstructions. The tool focusses on the precise annotation of metabolites with chemical structure representations. (Publication 1^[146]).
- Fast algorithms and implementations for processing chemical information are provided for the Chemistry Development Kit (CDK) library. This includes efficient algorithms for the identification of chemical rings (Publication 2^[147]), movement of electrons for Kekulisation and tautomerism, and handling stereochemistry (Publication 4, in preparation).

This dissertation is structured in two parts. Part I explores metabolic reconstruction alignment and annotation. Part II describes the cheminformatics technical aspects required for Part I.

1.5 Publications

This thesis is formed of four publications, at the time of writing, two have been submitted and accepted in peer review journals.

1. **May JW**, James AG, Steinbeck C. Metingear: a development environment for annotating genome-scale metabolic models. *Bioinformatics*, 29(17):2213-2215, 2013
2. **May JW**, Steinbeck C. Efficient ring perception for the Chemistry Development Kit. *Journal of Cheminformatics*, 6(3), 2014
3. **May JW**, James AG, Steinbeck C. Rapid and flexible alignment of metabolic reconstructions through chemical structures. *in preparation*
4. Willighagen EL, **May JW**, Berg AB, Jeliakvova N, Rahman SA, Rijnbeek M, Cherto MR, Spjuth O, Torrance G, Guha R and Steinbeck C. The Chemistry Development Kit (CDK). 3. Atom typing, Rendering, Molecular Formula, and Substructure Searching. *in preparation*

PART I

CHEMICAL DATA IN METABOLIC
RECONSTRUCTIONS

Chapter 2

Alignment of reconstructions

2.1 Introduction

2.1.1 Motivation

The creation of genome-scale metabolic reconstructions requires the integration of knowledge from different sources. Identifying when metabolites, reactions, or enzymes are equivalent allows the study of similarities and differences in metabolism. In creating reconstructions, it eliminates redundancy and enables the transfer of facts about these components. Facts that can be transferred include the presence of a component (metabolite, reaction, or gene) and annotations (reaction direction and lethality).

There are several levels of granularity at which reconstructions and their contents can be compared. The term reconstruction *alignment* shall be used in this chapter to describe the pairwise identification of equivalent metabolites. Since reactions are described as the transformations of metabolites, a correct alignment of metabolites is sufficient to obtain a reaction alignment. Differences in stoichiometry, compartments, and cofactors may optionally be considered.

2.1.2 Identifying metabolites

To understand how metabolites may be compared and aligned, we first consider how they are described. In a reconstruction, a metabolite may be described or annotated

with one or more of the following: internal identifier (or abbreviation), name, external resource identifier, formal charge, mass, molecular formula, and structural formula.

The internal identifier is either local to the reconstruction or a resource used in its creation. If a single resource is used, this may be sufficient for finding identical metabolites.

The name of a metabolite is usually trivial (or common) but may also be systematic. The use of different names for the same metabolite, means that names cannot be used for direct comparison. Conversely, the imprecise or ambiguous naming of different metabolites as the same is problematic.

As with the internal identifier, two metabolites with the same external identifier may be sufficient to determine equivalence. When metabolites are annotated to different external resources, a mapping is required to translate between them. Differences in content and scope mean direct translation may not be possible. An entry in one resource may map to several in another (Fig. 2.1).

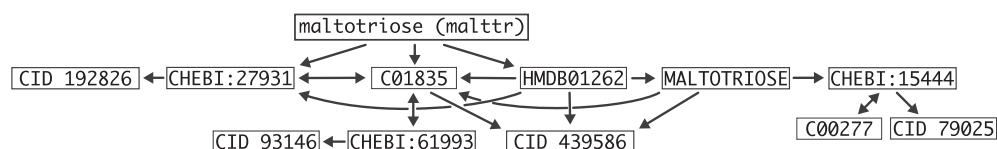


Figure 2.1: Direct and indirect cross-references to external resources for maltotriose ([malttr](#)) in Recon 2^[64].

The external database entry may be at a different protonation state or have undefined or inverted stereochemistry to that in the reconstruction. The metabolite is then misrepresented. For the maltotriose example, annotations are added for different stereoisomers (α/β), tautomers (ring-chain), and an erroneous link to a polysaccharide ((1 \rightarrow 4)- α -D-glucan).

The chemical properties such as formal charge, mass, and molecular formula can be the same for completely different metabolites and do not provide identification.

The structural formula provides a means of unambiguous identification. The structure is unique but differences in representation and precision again mean a direct comparison may not be possible. However, unlike external identifiers, the structure representations describe these differences and allow us to reason about the differences directly.

2.1.3 Existing approaches

Based on the information available, there are several approaches in which metabolites can be aligned.

Identifier mapping

Metabolites annotated to different external resources can be compared if a mapping between those resources is available. Given two resources, an identifier mapping provides pairs of identifiers that are identical. Maintaining all n^2 mappings can be impractical. Services or tools that provide a mapping will usually use a common primary identifier to which other resources are assigned.

Among others, identifier mapping services are: the Chemical Translation Service^[48], BridgeDB^[148], and UniChem^[149]. Tools for mapping identifiers were recently evaluated in expanding the annotation of Recon 2^[119].

Merged databases

Merged databases integrate entries from multiple child databases as a single unified (parent) resource. Entries from the child resource can then be aligned by inspecting the parent entries.

BKM-React^[50], MetRxn^[51], and MNXref^[52] are such databases specific to metabolism. The contents of these resources was outlined in the main introduction. All the resources primarily use the chemical structure for identifying equivalent metabolites but methodologies vary slightly.

BKM-React utilises the InChI and normalised metabolite names. The InChI ionisation layer is removed to allow matching of metabolites at difference protonation states.

In MetRxn metabolites and reactions are merged in an iterative process, using the chemical structure and balanced reactions to identify similar entries. Canonical and isomeric SMILES are computed for structures using Open Babel (via ChemSpider). The structures are normalised to pH 7.2 using JChem. Reactions with unresolved metabolites are iteratively refined through phonetic comparisons. MetRxn also undergoes some manual curation to improve coverage.

MNXref normalises the structure of metabolites using JChem to eliminate differences in protonation. InChI and SMILES are computed for each metabolite. The layers of the InChI are inspected and if reactions are found that encode metabolites with different stereochemistry, metabolites are kept separate, otherwise they are merged. For generic compounds that cannot be encoded in the InChI, SMILES is used. In both cases, polymers are compared with structural identity and formula checking. Metabolites with exact name and formula matches are also considered. Through reaction context, the evidence given by structural, name, or cross-reference is resolved through a majority vote. MNXref runs as a fully automated method.

Specific tools

Several tools are available for comparing or merging two reconstructions. The pairwise merging of reconstructions can be more sensitive than the merged databases. In a database, if one metabolite matches two entries, all three may be kept as separate records. In a pairwise comparison, one can merge or split entries to achieve a one-to-one mapping.

SemanticsSBML^[150] is non-specific to metabolic reconstructions and merges SBML entries through shared external identifiers. Annotations can also be propagated through inspection of connected elements in the network^[151]. ModeRator^[152] compares genome-scale reconstructions primarily using names. Formulae annotated on metabolites are used to indicate better comparisons. modelBorgifer^[153] compares reactions and metabolites through string similarity of multiple annotations and network topology. The scores of these attributes are weighted through machine learning.

The SuBliMinaL toolbox^[61] takes a more targeted approach and utilises relationships in the ChEBI ontology. This requires that metabolites are present in ChEBI. In a fuzzy match mode, metabolites that are found to have a common parent in the ontology can be merged.

2.1.4 Introduction to chemical identity

To utilise chemical structures in identifying equivalent metabolites, methods are needed to determine whether two structures are the same. Determining chemical identity has a long history in chemical registration systems, the key techniques are introduced here.

Modelling chemical structures as a graph, two structures are the same if there is an isomorphism that preserves the adjacency relation. That is, there is a bijection (or mapping) between the atoms such that the bonding in one structure is preserved in the other.

Isomorphisms can be located using a backtracking procedure that iteratively builds up the bijection^[154,155]. On modern computers, these algorithms can run very fast but still take exponential time in the worst case. To find one entry in a large collection of structures, the backtracking search must be run on each graph in the collection.

To allow for efficient retrieval in large structure collections, a canonical form can be used. A canonical form (or labelling) of a graph is a unique ordering of the vertices and edges. Comprehensive algorithms for the general case are still exponential but the canonical form can be precomputed and stored for efficient retrieval.

By far the most famous algorithm in chemical information processing is the Morgan algorithm^[156]. The Morgan algorithm computes a canonical label by iteratively exploring the connectivity of adjacent vertices (extended connectivity). Initially, one or more properties that are invariant (under reordering) are chosen for the vertices (atoms). Invariants are usually an integer such as the atomic number or degree. There will be several vertices that have been assigned the same invariant. The next step refines the initial invariants, a new invariant is assigned based on the current value and that of connected vertices. This process repeats and the connectivity information of each vertex propagates to adjacent vertices.

The refinement stops when either all labels are different or no new labels are being discovered. If there is still more than one invariant with the same value, a unique label can be assigned through a tie breaking step. A full list is beyond the scope of this introduction but popular tie breaking methods include traversing the graph from the lowest label and artificially modifying one invariant and rerunning refinement^[102].

The SEMA (stereochemically enhanced Morgan algorithm) introduced the ability to encode stereochemistry^[157]. The algorithm generates a SEMA name of variable length, a fixed size name can be produced through use of a hash function. This SEMA key can be used to efficiently search large collections for duplicates, optionally including stereochemistry^[158].

Through uniquely ordering the atoms and bonds as a canonical form it is possible to generate a unique (canonical) SMILES string^[102]. As a linear notation, it can be used

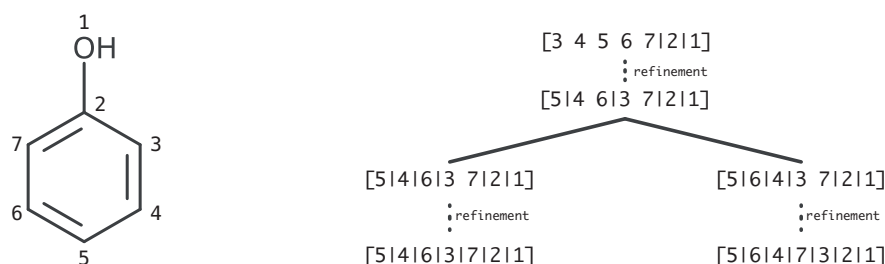


Figure 2.2: Phenol has one plane of symmetry (an automorphism). Initially, atoms are partitioned and sorted by atomic number and connected (heavy) atoms, only atoms 1 and 2 are distinct. Through refinement, the atoms 3 and 7 are adjacent to 2 and can therefore be distinguished from 4, 5, and 6. Similarly, atoms 4 and 6 are then distinguished from 5 as they are adjacent to 3 or 7. No further refinement produces a discrete labelling and so an atom from the lowest ranked set (4 or 6) is favoured to break the tie. In this case 4 and 6 are actually equivalent and favouring either produces an equivalent labelling.

as a structure key by encoding with a string hashing algorithm.

The described labelling algorithms are highly dependent on the initial invariants, and these invariants may be insufficient to differentiate non-equivalent vertices. Tie breaking procedures that expect all identical labels to truly be equivalent will generate more than one (non-unique) labelling. This can be overcome by using better invariants or by systematically generating every possible ordering (Fig. 2.2). Better invariants are likely to be sufficient for most chemical structures but the systematic generation guarantees a unique labelling. Examples of general graph algorithms that inspect all labellings are *nauty*^[159] and *traces*^[160]. By inspecting the symmetries of the graph (automorphisms) the algorithms can avoid generating every possible labelling.

Another alternative is to generate a key that does not depend on a unique labelling. This still allows for quick equivalence (or rather non-equivalence) checking but is not susceptible to generating different keys (no false negatives). Extended connectivity hash codes capture the environment of each atom that can be combined to a single structure key. The distinction here is the hash code is obtained directly and is not a digest of a canonical form. CACTVS hash codes^[161] are generated this way and shall be explored more in this chapter.

In recent years, the InChI has gained popularity as a free and open encoding for identifying structures and connecting databases^[104]. The canonical labelling algorithm used by the InChI is derived from *nauty* and encodes stereochemistry as a separate parity vector^[162]. A significant feature of the InChI is the layers, which allow comparison of different properties. Each layer is partially dependent on the previous and must be

peeled off sequentially. Note that the ionisation layer appears before stereochemistry. To account for representation differences, several standardisations are applied that modify the input structure^[163]. The InChI is therefore not a replacement for existing representations but an addition to assist in finding and linking information^[107].

Similar to the SEMA key, the InChIKey encodes the graph constitution and stereochemistry separately. InChIs can be generated from a molfile or CML using a binary executable; alternatively the API can be called programmatically. InChIs generated through the API may be different to those generated from the binary (Apdx. A.1).

The NEMA (newly enhanced Morgan algorithm) extended the SEMA method with more reliable stereo recognition and support for polymers^[164]. The NEMA can also distinguish mixtures and relative stereochemistry.

2.1.5 Objectives

Precise alignment of reconstructions is beneficial in their creation and comparison. If metabolites are indirectly annotated to the same resource, they can be compared through inspection of this common reference. If metabolites are annotated to different resources, it is common to create or use either an identifier mapping or merged database.

Although convenient, identifier mappings and merged databases must be created ahead of time. These common references can only account for the content included at their creation (or subsequent update). As new releases of child databases are made, their contents may be: deleted, merged, split, or modified. This can be difficult to capture and the contents of the common references may be out of date and contain errors. These resources are extremely valuable but on-demand merging of reconstructions can be more perceptive to subtle differences and can produce a better alignment.

The approach used by the SuBliMinaL toolbox to allow non-exact merging is powerful but depends on precomputed and potentially incomplete relationships in the ChEBI ontology. The ChEBI ontology encodes many relationships but only some of these are useful to consider for merging. For example, many entries have the parent *primary alcohol* (CHEBI:15734).

When considering what is feasible to merge, the relationships are: stereoisomers, conjugate acids and bases, tautomers, ions (oxidation), and isotopes. A specific challenge

is presented by carbohydrates that can exist and interconvert as either a chain (aldehyde) or ring (pyranose or furanose) form. The ring forms also exist as either α or β anomers. By completely disregarding all stereochemistry, metabolites that should be kept as different would be incorrectly merged.

Using reconstructions annotated with chemical structure representations, this chapter describes and explores the cheminformatics methods towards on-demand alignment of reconstructions. Connectivity hash codes provide an efficient way of checking whether two structures are potentially equivalent and are explored for aligning reconstructions. The hash codes are independent of representation and allow equivalent encoding of CTfile, SMILES, InChI, or CML annotations. The hash code is evaluated on chemical databases through checking for collisions.

Similar to the InChIKey, a more or less specific encoding can be used to find an exact of similar structure. The discrimination and performance of different specificity hash codes are measured on metabolic resources. Like the InChIKey, this only provides an all-or-nothing match and closer inspection is needed for a precise alignment. A candidate retrieval technique is introduced that identifies and categorised matches based on chemical structure. The retrieval is evaluated on matching reconstructions and a database; retrieved candidates are compared to existing gold standard mappings.

2.2 Methods

2.2.1 Connectivity hash codes

The extended connectivity hash code algorithms described by Ihlenfeldt W and Gasteiger J^[161] (used in CACTVS) have been implemented and extended. Several iterations led to the final version, issues found during evaluation will be discussed in the results but the resolutions are described here. The full source code of the final implementation is available in the Chemistry Development Kit. Source code listings for Java are included to aid description.

Preliminaries

Key to any hashing algorithm is a pseudorandom number generator (PRNG). Given a numeric value (or seed) a PRNG will generate a seemingly random number. When the numbers are continuously generated they should follow a uniform distribution. This uniform distribution ensures the full numeric range is utilised during encoding.

A fast random number generator suggested in the original publication is XOR shift. A new number is generated by XOR-ing with a bit shifted version of itself by chosen offsets^[165] (Listing 2.1). Although better PRNGs exist, the elegance and simplicity of the implementation is sufficient for encoding chemical structures. A useful property is that 0 is a fixed point and never generated otherwise.

Listing 2.1: 64-bit XOR shift pseudorandom number generator

```
long xorshift(long x) {  
    x ^= (x << 21); x ^= (x >>> 35); x ^= (x << 4);  
    return x;  
}
```

A number is *rotated* by sequentially generating pseudorandom numbers a specified number of times (Listing 2.2).

Listing 2.2: Rotating a number

```
long rotate(long x, int n) {  
    while (n-- > 0)  
        x = xorshift(x);  
}
```

```
    return x;
}
```

A number is *distributed* by rotating based on the input value (Listing 2.3).

Listing 2.3: Distributing a number

```
long distribute(long x) {
    // rotates 1 to 8 times depending on low order bits
    return rotate(x, 1 + (x & 0x7));
}
```

Combining values

To combine values, the XOR operator is again used as it is both associative and commutative. The operator cannot be used directly as combining the same value an even number of times cancels out. To avoid this, the number of times a value has been included (occurrences) is used to rotate the number before inclusion. This shall be referred to as occurrence count rotation.

Initial invariants

The initial step of the algorithm requires the generation of an invariant (or seed) for each atom. This is a numeric value that describes some attribute of an atom or its environment (e.g. element, charge, number of hydrogens, etc.). The original publication assigned ranges of the prime number table to predetermined properties. This was used to guarantee uniqueness of the initial invariants.

This implementation encodes attributes using a functor. The functor takes an atom (and the structure it belongs) and generates a numeric value (Listing 2.4). Multiple encoders are combined with an initial prime and are distributed after each inclusion. This approach is modular and allows extension to encode user-defined properties.

Listing 2.4: Functor interface to encode atom attributes

```
interface AtomEncoder {
    long encode(Atom a, AtomContainer ac);
}
```

Refinement

An initial invariant is generated for each atom in a structure. The invariants are then augmented through information on neighbouring atoms in a process similar to the Morgan algorithm. At each iteration an atom feels the influence of the next sphere of connected atoms. After each atom has been updated, the new values are copied back to the invariants and the process repeats. The output of the refinement is a vector of invariants that characterise the environment of each atom.

Unlike the Morgan algorithm, the number of iterations is bounded by a fixed *depth* parameter. If the depth value is too small, structures with long chains of varied length (e.g. lipids) may be encoded as equivalent. Using a larger value takes more time to compute, the original publication suggests a depth of 32.

At each stage, the invariants of the neighbours are combined with current value using the occurrence count rotation. A naive check of all values for occurrence is appropriate. Although this has unfavourable $O(deg(v)^2)$ complexity, it is fast because $deg(v) \leq 4$ for the majority of organic atoms.

Stereochemistry

Initially stereochemistry was included through assigning Cahn-Ingold-Prelog (CIP) labels as initial invariants. The labels were used due to familiarity and because they were already partially provided in the CDK codebase. Improving the CIP labelling was investigated (Chap. 6) but still found to be too slow.

An alternative described by original publication is to encode stereochemistry using the order of invariants. The CIP labelling is an ordering of neighbours around a stereocentre. Rather than using the ordering defined by the CIP rules, the ordering of the adjacent invariants is used. Stereochemistry is checked after each iteration during refinement. If at some stage all the adjacent invariants are different, a numeric label for the stereocentre is assigned and the invariants updated. The label is determined by the permutation parity of winding and ordering^[166].

In this implementation, stereochemistry is handled by a compossible StereoEncoder (Listing 2.5). After each refinement step, the current invariants (*curr*) and next invariants (*next*) are provided to the encoder. An encoder then checks whether stereocentres have a configuration (based on the current invariants). If a configuration is found, the

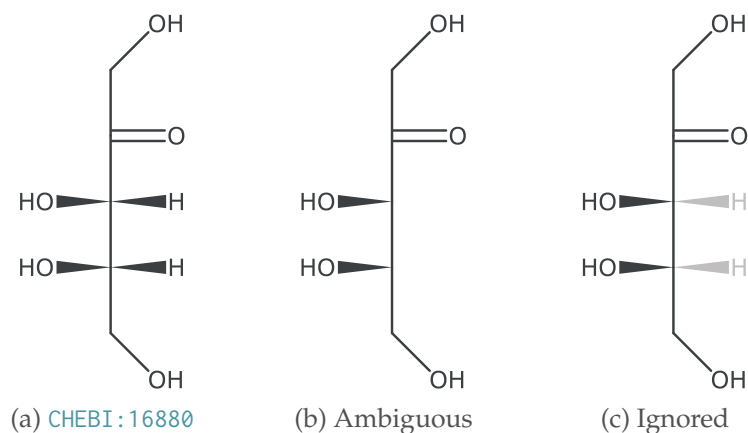


Figure 2.3: Removing hydrogens attached to chiral atoms may result in ambiguous stereochemistry. By ignoring the hydrogens, the correct stereochemistry can be encoded. The input structure also remains unmodified.

encoder modifies the value for the stereocentre atom in `curr` and places it in `next`. Two arrays are needed to ensure stereoencoding is independent of atom order.

There are different stereo encoders for each type of stereochemistry. Each encoder is composed into a single StereoEncoder that is used in the refinement. This allows certain stereochemistry (e.g. geometric) to be disabled whilst still encoding others (e.g. tetrahedral). It also allows new stereochemistry to be encoded and defined as needed.

Listing 2.5: Interface to encode stereochemistry

```
interface StereoEncoder {
    boolean encode(long[] curr, long[] next);
}
```

The hash code uses both the depiction and stereochemistry data structures. Structures loaded from different formats (SMILES, InChI, or molfile) obtain the same hash code with stereochemistry. Chapter 6 describes more details on the handling of stereochemistry. The current version of the hash code can distinguish: tetrahedral, geometric (double-bonds), and cumulated (extended tetrahedral and extended geometric) stereochemistry.

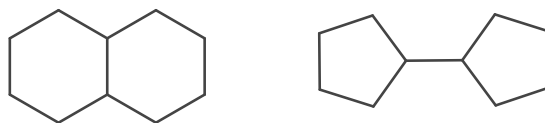
Suppressed atoms

A structure may include hydrogens as explicit vertices in the graph, counts on each atom, or a mixture of these. Removing hydrogens as a preprocessing step is trivial but may cause problems, particularly around stereocentres (Fig. 2.3b). Rather than actually removing the hydrogens, the implementation simply ignores them during value combination. Atoms are ignored by setting the value to 0; recall 0 is a fixed point for the XOR operations and the XOR shift PRNG.

The coordinates of the hydrogen are still used in allowing correct interpretation of stereochemistry (Fig. 2.3c). Atoms other than hydrogens can also be ignored.

Perturbation

The refinement procedure outlined previously is insufficient in discriminating even simple cases. The refined invariants that describe the atom environments in these two structures are identical:



Although false positives are expected by chance, the collision here is due to a deficiency of the refinement step. The original publication introduces several approaches to tackle these cases, one of which is to systematically perturb invariant values to break symmetries. This is again similar to canonical labelling; however only one round of symmetry breaking is undertaken.

After the initial refinement has finished, a set of atoms that have been assigned an equivalent invariant value are selected. The invariant value of each atom is then rotated and refined. This generates several sets of environments for each atom. Once all atoms have been individually perturbed and refined the different environments of each atom are combined using occurrence counting.

The choice of which atoms to perturb affects the fidelity of discriminating equivalent vertices. The original publication described using the smallest set of atoms that were non-recursively removable from the structure. This set includes atoms in and between ring systems. Only atoms in rings are required but the original publication does not perform a full ring perception due to increased effort.

In this implementation only atoms in rings are used. This was possible due to the improved ring perception (Chap. 4). Selecting a smallest set of equivalent atoms in a ring to perturb was still found to be insufficient to discriminate all symmetries. To this end, several sets of equivalent atoms were tested. In the results, these will be referenced by level:

Level	Atom set
minimal	a smallest set of ring atoms (default)
intermediate	all smallest sets of ring atoms
maximum	all ring atoms

Molecule and ensemble hash codes

The algorithms introduced above provide a vector of invariants that characterises the environment experienced by each atom in a structure. To generate a single *molecular hash code*, the invariants are sorted and combined into a single value. These atoms can be combined using the occurrence count rotation. When sorted, invariants of the same value appear sequentially and their frequency can be easily counted.

In a similar manner, the original publication describes combining the hash codes of multiple structures into an *ensemble*. This process can be extended even further to encode an *ensemble of ensembles*, representative of a reaction.

Implementation

The hash code API is modular allowing inclusions and extension with new atom attributes or types of stereochemistry. A fluent API provides intuitive creation of a hash generator with specified parameters (Listing 2.6). The `MoleculeHashGenerator` is a functional interface; when provided a structure it produces a single 64-bit value.

2.2.2 Candidate retrieval

Like the InChI layers, a more or less sensitive hash code can be generated depending on the chosen parameters. The hash code can then be used to find a non-exact match when a more specific match is not available. Two methods were used for finding and retrieving a candidate match.

Listing 2.6: Creating a hash encoder using the fluent API

```
MoleculeHashGenerator hashGen = new HashGeneratorMaker().depth(32) // #: refinement depth
    .elemental() // E: atomic number
    .chiral() // S: stereochemistry
    .charged() // C: formal charge
    .isotopic() // I: mass number
    .perturbed() // P: perturbed hash codes
    .molecular(); // the type of hash code

// generate a key for each structure in a collection
for (IAtomContainer ac : acs) {
    long key = hashGen.generate(ac);
}
```

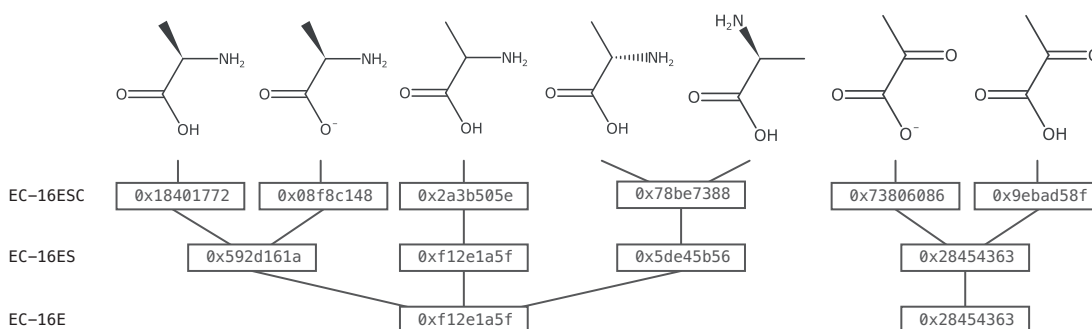
Bottom-up retrieval

Figure 2.4: Structures in a reference are first indexed by a specific hash code (ec-16ESC). If no match is found when queried, a less specific hash code is calculated for all entries (ec-16ES). This continues until a match is found or all levels have been explored.

Initially metabolites were retrieved using a bottom-up approach (Fig. 2.4). Metabolites are first indexed with a high level of detail. If no exact match is found, a hash code encoding less specific details is calculated. Descending to a less specific match requires all structures at that level are indexed. This approach is useful when structures are stored for later retrieval. The search stops when a matching hash code is found, and all structures with that value are returned.

Top-down retrieval

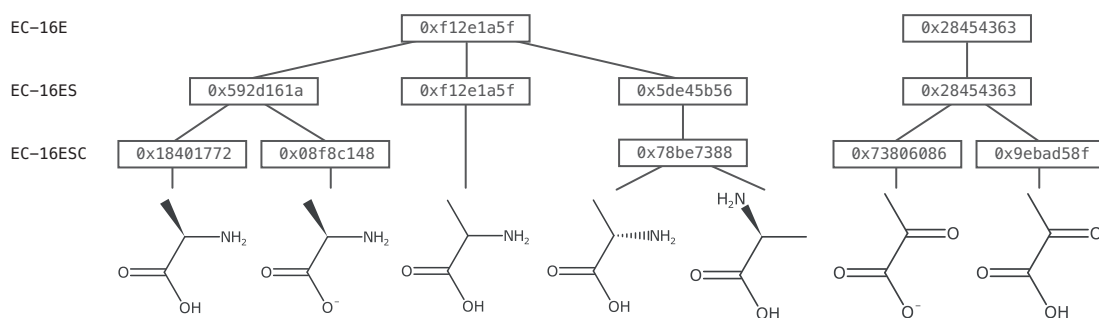


Figure 2.5: Structures in a reference are first indexed by a loose hash code (ec-16E). If no match is found, the search is complete. If a unique match is found, it is returned straight away. If a non-unique match is found, more specific hash codes (e.g. ec-16ES) are calculated in an attempt to refine the match and provide a single candidate.

A more efficient approach for on-demand retrieval is to find a match with a top-down search (Fig. 2.5). This effectively inverts the search tree but has advantages. Hash codes encoding more detail are more expensive to compute and these are now only generated for a subset of structures. If a discrete entry is found, a more detailed hash code will not refine the match and can be skipped. The search stops when a single or no matching hash code is found. When no match is found, the search backs up and returns all entries at the previous level.

2.2.3 Candidate reasoning and classification

The retrieval based on hash codes provides a set of candidate matches. These candidates are then processed further with more intense calculations to quantify the structure based match.

Standardisation

The following standardisations are applied (in order) to each candidate after it has been retrieved.

1. Explicit hydrogens that can be suppressed are converted to a labelled count on the attached atom (implicit). A hydrogen atom can be suppressed if it has no

formal charge, mass label, and exactly one non-hydrogen neighbour. Data structures representing stereochemistry are also updated (Chap. 6).

2. Terminal oxygen, nitrogen, and sulphur atoms with a formal charge are neutralised by adding or removing protons. A more comprehensive approach can be used but this method was found to be sufficient.
3. Metal atoms (as defined by the InChI^[163]) are disconnected. This is solely for normalising porphyrin and corrin coordination complexes and a more targeted approach could be adopted.
4. A unique tautomer is generated (Chap. 5).

Isomorphisms

Isomorphisms between two structures are calculated using the backtracking Vento-Foggia (VF) algorithm^[155]. The implementation lazily generates mappings between the atoms of two structures. There may be more than one mapping per structure. These are represented as a permutation of the query vertices (Tab. 2.1). When generating the mappings, only the element type is checked for compatibility. Hydrogens, formal charges, and bond orders are not used to allow the mapping between structure with different saturation and tautomers.

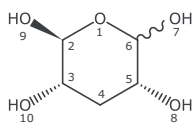
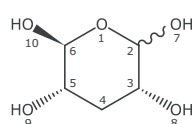
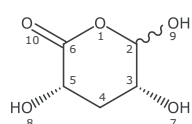
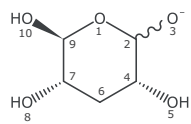
Scoring

For each retrieved candidate, the isomorphisms are found and four parameters are counted. A parameter is incremented when the atoms in both structures have the same value. The parameters are:

1. v , the bonded valence, including bonds to implicit hydrogens
2. x , the connectivity, including bonds to implicit hydrogens
3. s' , stereo mismatch, number of stereocentres with inverted configuration
4. s , stereo match, number of stereocentres with the same configuration

The stereo match and mismatch are determined through comparing normalised data structures (Chap. 6). Geometric isomers (double bond stereocentres) are only counted once. The stereo score is split into two separate values to allow classification of unspecified stereochemistry.

Table 2.1: Examples of isomorphisms between a query and target structure represented as a permutation of the query vertices in cycle notation (Apdx. A.2). All bond orders and stereochemistry is ignored allowing two matches for each pair.

Query	Target	Permutation
		$(2,6)(3,5)(9,10)$ $(7,9,8,10)$
		$(2,6)(3,5)(7,10,9)$ $(7,9,8)$
		$(3,4,6,9,8,5,7)$ $(2,9,5,4,6)(3,7,10)$

The scores of each mapping are compared in the order listed above and the best mapping is selected. When ordering, the stereo mismatch is negated so mappings with no inverted stereochemistry appear first.

Categorisation

Candidates are categorised by inspecting the values computed in the scoring. The following table describes how categorise are decided:

Condition	Category
$v < V $ or $x < V $	skeleton
$s' = 0$ and $s = s_{query} \cup s_{cand} $	identical
$s' > 0$	diastereoisomer
$s < s_{query} $ and $s < s_{cand} $	unspecified
$s = s_{query} $ and $s < s_{cand} $	less specific
$ s_{cand} - s = 1$	less specific (single)
$s < s_{query} $ and $s = s_{cand} $	more specific

Where:

- $|V|$ is the number of explicit atoms
- $|s_{query}|$ is the number of stereocentres with configuration in the query structure
- $|s_{cand}|$ is the number of stereocentres with configuration in the candidate

Elimination

When more than one candidate is retrieved an elimination procedure is used to remove worse scoring candidates.

1. If one identical candidate is found, all non identical candidates are removed.
2. If one candidate has $s' = 0$, all candidates with $s' > 0$ are removed.
3. If one candidate has $v = |V|$ and $x = |V|$, all candidates where $v < |V|$ or $x < |V|$ are removed.
4. If one candidate has $s' = 0$ and $s > 0$ all candidates where $s = 0$ are removed.

2.3 Results and discussion

Hash code parameters

The extended connectivity hash codes have varied parameters that can be optionally included. When referring to the hash code implementation the prefix, *ec-* is used. Similar to the FICTS system used by original CACTVS hash codes, characters are used to describe the configuration.

#	depth	number of refinement steps
E	Elemental	atomic number
S	Stereochemistry	tetrahedral, geometric, and cumulated configurations
C	Charged	atom formal charge
I	Isotopic	mass number
B	Bond orders	bond orders
P	Perturbed	break symmetries

The encoding “*ec-8ESC*” represents that a depth of eight was used to encode differences in elements, stereochemistry, and charge.

External resource identifiers

Throughout this results section, identifiers are included to external resources. A guide to these identifiers is provided in [Appendix A.3](#).

2.3.1 Exploratory analysis of aligning *iJR904* and MetaCyc 16

To test the feasibility of using hash codes to align metabolites, the *iJR904*^[167] reconstruction of *E. coli* and the MetaCyc (version 16) database was utilised^[34].

The *iJR904* reconstruction was imported and annotated with Metingear ([Chap. 3](#)) and aligned using the bottom-up comparison against MetaCyc. The version of the hash code used could optionally encode, tetrahedral stereochemistry, formal charge, and bond order. Stereochemistry was encoded using an atom labelled descriptor (CIP). Metabolites of *iJR904* were annotated through matching metabolite names to ChEBI

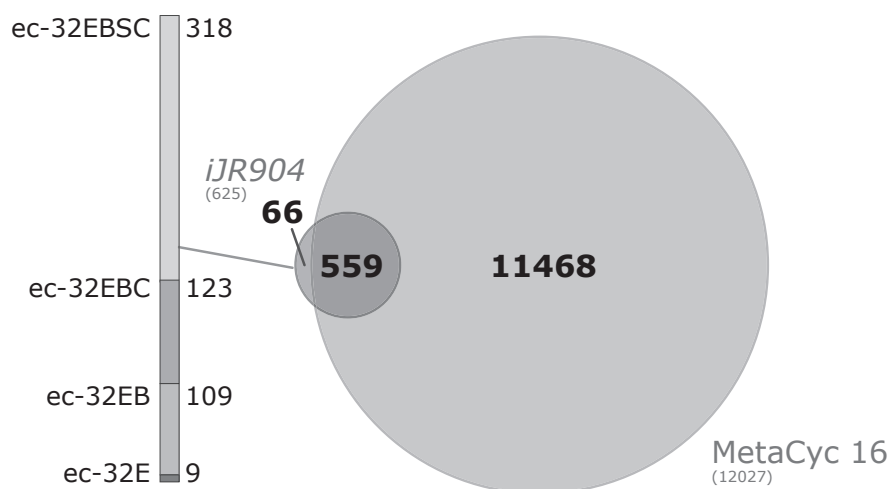


Figure 2.6: *iJR904* metabolites that matched at least one entry in MetaCyc 16. Of the 559 that matched, a breakdown shows at what hash code level a match was found.

and transferring the chemical structure in Metingear. Structural annotation was not added for 53 metabolites.

Using just the hash codes, 559/625 metabolites from *iJR904* were paired with at least one structure in MetaCyc (Fig. 2.6).

Of the 559 metabolites a match was obtained at various specificity of hash code (Fig. 2.6). The majority of entries (318) matched when charge, stereochemistry and bond order were considered.

iJR904 was partially created from EcoCyc (a subset of MetaCyc) and metabolite name should be relatively consistent. By checking names and synonyms 83/559 metabolites were found that matched using the hash code but not with an exact string comparison (Fig. 2.7).

There were 24 metabolites that were correctly matched but had either completely different names or minor stylistic differences (e.g. beta, B, or β). Entries that were matched one-to-many were not counted as a name difference.

The *iJR904* reconstruction contains metabolites with non-specific stereochemistry but MetaCyc only had discrete entries. A match is then only obtained without matching stereochemistry hash codes and more than one MetaCyc entry is identified. MetaCyc contained structures with stereochemistry depicted as a projection. Projections such as Fischer and Haworth are common in depicting carbohydrates, they are easy for a hu-

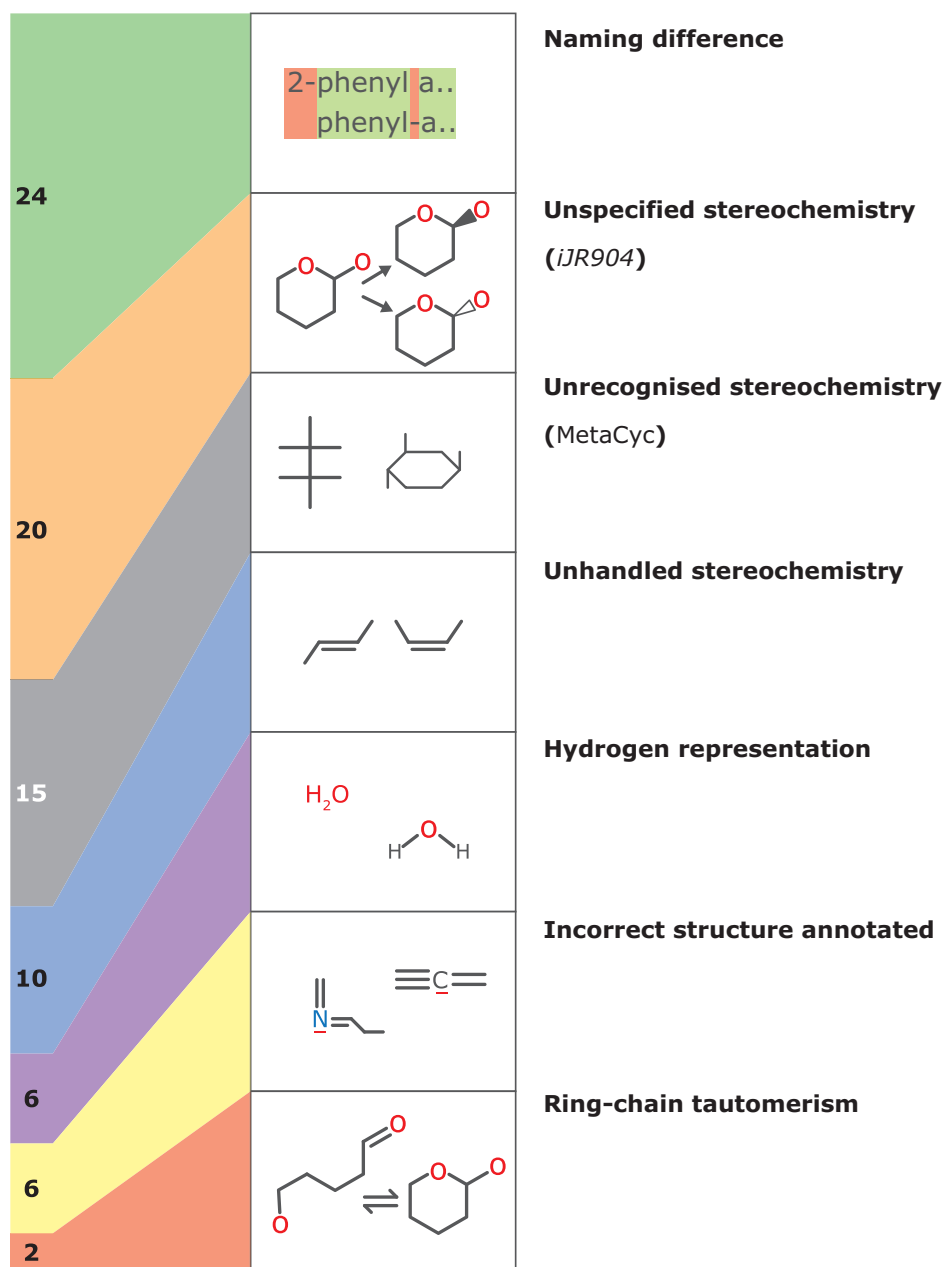


Figure 2.7: Matches obtained with the hash code were tested for matching names. Each entry that did not match by name but did by hash code (83) was manually inspected. Naming differences are where a single correct match was found but the names differed. The other categories retrieved multiple matches in which one or none may be correct.

man to interpret by require specialised computational handling. This can sometimes lead to missing stereochemistry and resulted in multiple entries being found.

The hash code implementation only had limited support for encoding stereochemistry. Geometric isomers and structures with rotational symmetry could not be distinguished. Hydrogens may be represented as either a labelled count or an explicit atom, hydrogens represented as explicit were removed before generating a hash code. This hydrogen removal also removed the stereochemistry (Fig. 2.3) resulting in multiple matches being retrieved.

Metabolites were annotated with the wrong structure in both the reconstruction and the database. Entries in *iJR904* were automatically annotated from ChEBI leading to mistakes due to imprecise names. Incorrect structures in the reference had incorrect stereochemistry or functional group attachments and a different match was found (Apdx. A.4). These were reported to MetaCyc for correction. Finally, two carbohydrates in *iJR904* were annotated with chain-forms and had matched a different entry in MetaCyc.

With the exception of the last three categories, a correct or acceptable match was identified but sometimes only as a one-to-many relationship. A refined hash code implementation was created with improved encoding of stereochemistry and different hydrogen representations (see Methods). The general handling of stereochemistry in CDK was also improved (Chap. 6).

2.3.2 Hash code collision analysis

The exploratory analysis identified several technical issues with the hash code, in particular stereochemistry encoding. Having resolved these issues the hash code was evaluated on ChEBI (Sept 2012) and PubChem-Compound (Dec 2012). Both databases are non-redundant and were used to identify any collisions. A detected collision may be due to chance, algorithm deficiency, or actual duplicate entries. A hash code that encoded all available attributes (e.g. ec-32ESCIBP) was used.

The ChEBI database was chosen as a curated resource containing entities of biological interest (inc. metabolites). PubChem-Compound was chosen as it is the largest openly available chemical database and would highlight any corner cases.

ChEBI

Collisions in ChEBI were observed for six reasons (Tab. 2.2).

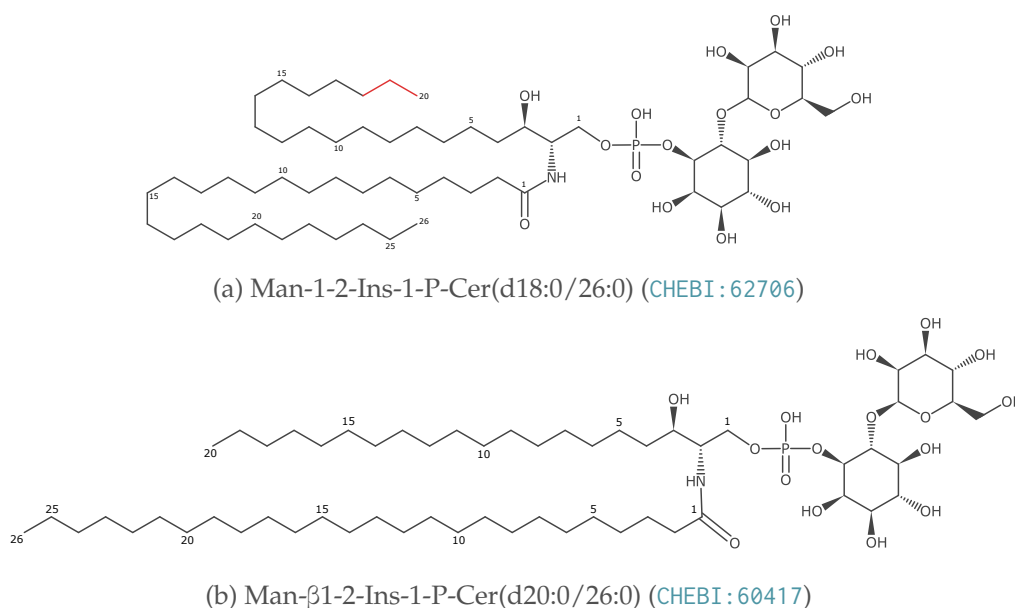


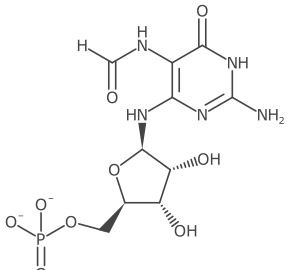
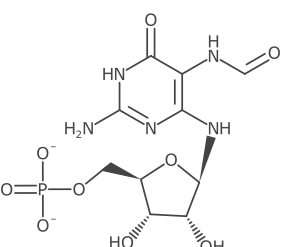
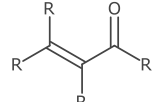
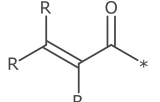
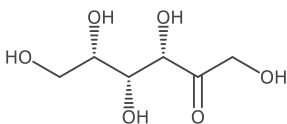
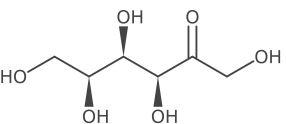
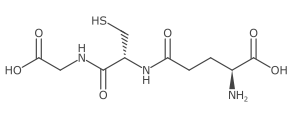
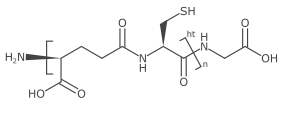
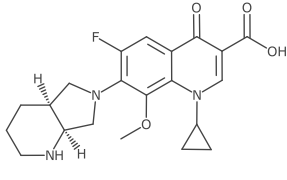
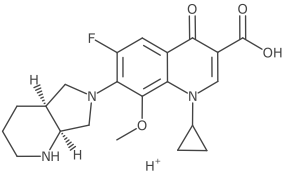
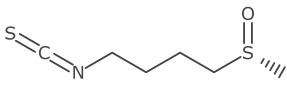
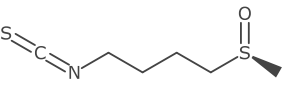
Figure 2.8: The hash code identified these two entries as identical. The depiction for CHEBI:62706 incorrectly contained two additional atoms (shown in red). These atoms have now been removed.

Three pairs of duplicate entries were found (Tab. 2.2a). Two of the pairs (CHEBI:57258 and CHEBI:59548, CHEBI:17038 and CHEBI:52361) were merged into single entries. The other pair was found to be an incorrect depiction that has now been updated by ChEBI curators (Fig. 2.8).

Pseudo atoms (non-periodic element) are included to represent variable attachment points, aliases, or labelled groups (Tab. 2.2b). The alias atom ('*') and R group ('R') are used in ChEBI to distinguish between functional groups and common substructures. Elements in the hash code are distinguished through the atomic number, which is zero for both 'R' and '*'. If required, these structures can be hashed differently by defining a custom functor that encodes the label.

Identical structures were found for ontology parent and child classes. The structure of L-sorbose (CHEBI:17266) was identical to a child entry, keto-L-sorbose (CHEBI:13172) (Tab. 2.2c). Structures for the parent classes have now been updated or removed.

Table 2.2: Examples of ChEBI entries identified as equivalent using the hash code.

(a)	 CHEBI:57258	 CHEBI:59548	true duplicates
(b)	 CHEBI:51689	 CHEBI:23916	pseudo atoms
(c)	 CHEBI:13172	 CHEBI:17266	ontology classes
(d)	 CHEBI:16856	 CHEBI:60836	polymers
(e)	 CHEBI:63611	 CHEBI:63699	unspecified protonation
(f)	 CHEBI:47808	 CHEBI:47809	unrecognised stereochemistry

The CTfile Sgroups are a supplementary (substructure) description on top of the base structure. The CDK library does not interpret these and the structures of glutathione (CHEBI:16856) and phytochelatin (CHEBI:60836) are seen as identical (Tab. 2.2d).

Unspecified protonation is represented in ChEBI by a disconnected proton atom. These are suppressed during hash code generation producing the same encoding for moxifloxacin (CHEBI:63611) and [moxifloxacinium]⁺ (CHEBI:63699) (Tab. 2.2e).

The encoding of stereochemistry had been improved but some rarer cases were still not recognised. Tetrahedral stereochemistry involving a lone pair of electrons ((*R*)-sulforaphane (CHEBI:47808) and (*S*)-sulforaphane (CHEBI:47809)) was not encoded as the implementation required four neighbours (Tab. 2.2f). Other unencoded stereocentres included extended tetrahedral (CHEBI:38401) and octahedral (CHEBI:36409) configurations. After discovering these collisions, the implementation was updated. Normally involving organometallics, representing octahedral configurations is difficult and are still not encoded.

PubChem-Compound

Hash codes for ~45 million entries in PubChem-Compound (Dec 2012) were calculated and compared. Using the minimal level of perturbation, 58 pairs of compounds obtained the same hash code (Fig. 2.3). No collisions by chance were found and inspection revealed the collisions were due to a deficiency in the perturbation step. The pairs of entries had the same formula but different connectivity. The perturbation was extended to capture the observed differences. The three levels are *minimal*, *intermediate*, and *maximal* (see Methods).

With intermediate perturbation, only 4 pairs of entries were encoded as the same hash code. Maximum perturbation was sufficient to encode a unique key for all entries in PubChem-Compound. The maximum perturbation is a lot more computationally intensive and somewhat negates the benefit of using a hash code. Therefore the default option is to encode using the minimal perturbation.

In PubChem-Compound, similar collision examples were not found as in ChEBI because of automated standardisation and merging. For example, polymers (Tab. 2.2d) are combined into the single CID 124886 entry. Likewise, entries with unhandled stereochemistry (e.g. octahedral) are also assigned to a single record.

Table 2.3: The minimal level of perturbation was found to be insufficient to distinguish pairs of structures in PubChem-Compound. These structures generated the same hash code, but not by chance.

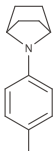
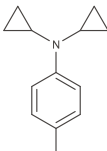
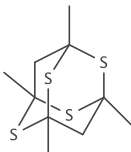
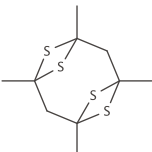
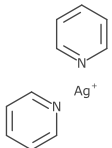
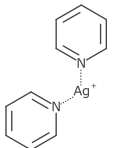
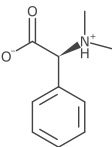
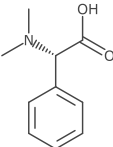
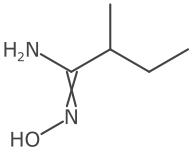
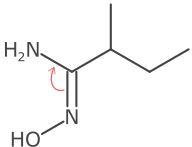
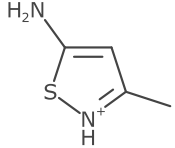
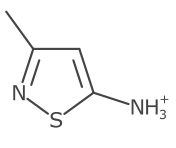
Example		Perturbed	n pairs
 CID 44333798	 CID 57170558	minimal	58
 CID 138898	 CID 241107	intermediate	4
-	-	maximum	0

Table 2.4: Classification of false negatives compared with InChIKey. Some ignored geometric isomers were also categorised as mesomers and the counts are therefore approximate.

Example	<i>n</i> sets	Reason
 CID 50912158	 CID 4237302	17,199 metal containing
 CID 38988795	 CID 38988796	85,254 zwitterionic tautomer
 CID 4643229	 CID 18619918	~23,028 ignored geometric isomers
 CID 24848379	 CID 1268123	~9,911 mesomers

PubChem-Compound false negatives

InChIKeys were calculated (Apdx. [A.1](#)) for all PubChem-Compound entries and compared to the generated hash codes. This was performed to highlight potential false negatives. A false negative here is when different hash codes are encoded for structures identified as equivalent by the InChI. There were 280,940 entries (~0.6%) that were found to share an InChIKey with at least one other entry. Manual inspection and automatic categorisation highlighted four reasons for the differences (Tab. [2.4](#)).

All cases were seen to be due to standardisation performed by the InChI. Dipolar bonds to metals are disconnected. Charges are neutralised and standardised allowing equivalence of mesomers and zwitterionic tautomers. Double-bond stereocentres responsible for geometric isomerism are ignored if the bond is identified as tautomeric or resonant.

With the exception of the metals, the same hash code could be obtained for the other pairs by disabling the encoding of charge or stereochemistry.

2.3.3 Structure key distributions

The collision analysis demonstrated the ability of the hash code to discriminate all entries in PubChem-Compound.

Hash codes can be calculated for different levels of sensitivity allowing a variable match. A more sensitive key takes longer to compute but may not necessarily discriminate more entries. To explore this the hash code and additional structure keys based on: counts, element frequency (formula), and the standard InChIKey were tested.

The discrimination ability of different keys (Tab. [2.5](#)) was examined on metabolite structures from: SEED (May 2013), MetaCyc (version 16), and MNXref (June 2013). These databases are specific to metabolism and representative of the contents of reconstructions. The keys for every structure in each database was calculated. The entries were then grouped (in bins) by key. The number of structures that were assigned each key was counted. When more structures are encoded as a single unique key, the discrimination is better. Figure [2.9](#) shows the frequency distribution for bins of size 1-10.

Table 2.5: Evaluated structure keys.

Key	Category
$ V $	Atom ($ V $) and bond ($ E $) counts
$ E $	
$ V , E $	
formula-h	Element frequency, including and excluding hydrogens
formula+h	
ec-4E	Extended connectivity hash codes
ec-4ES	
ec-16ES	
ec-32ESP	
ec-16ESCIP	
InChIKey	Standard InChIKey

As expected, keys based only on the atom and bond counts were not able to discriminate well; many entries had the same value. A key based on the molecular formula was able to completely separate $\geq 20\%$ when hydrogens were excluded and $\geq 40\%$ when they were included. Encoding only elements and their extended connectivity to depth of 4 provided slightly more separation (55%-60%) but could be increased to (75%-80%) when stereochemistry was encoded. Increasing the depth and perturbing the hash code had only minimal impact. The InChIKey does not reach 100% due to pseudo atoms. Structures obtained the same value for ec-16ESCIP were because bond orders were not encoded. In the case of MNXref, polymers are encoded the same as a monomer, this is also seen in the InChIKey distribution.

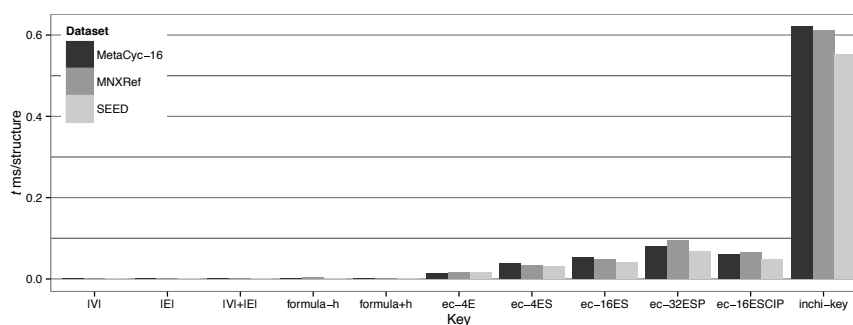


Figure 2.10: The average time take to encode an entry with different structure keys for each dataset.

The time taken to encode the different keys was also measured. Reported times are for a Intel Core i7 CPU (2.66 GHz) with 8 GB of RAM.

The direct access values of atom and bonds counts are computationally very cheap.

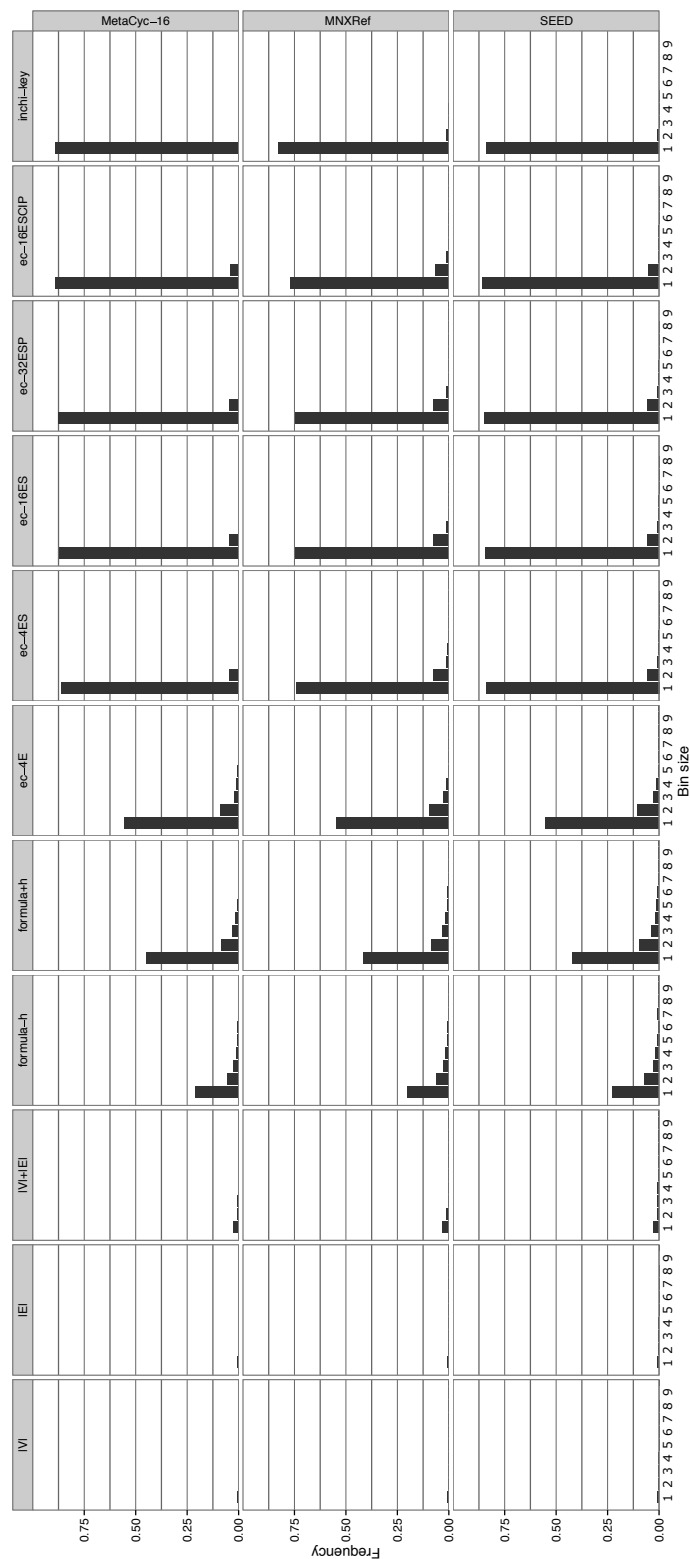


Figure 2.9: Frequency distribution of different structure keys. The more structures assigned a unique key, the better the key is at discriminating entries. The connectivity hash codes were not encoding bond orders, accounting for the structures with non-unique keys (Bin size ≥ 2). The InChI does not reach 1.0 because the current version does not encode pseudo atoms.

The element frequency keys are derived from the structure (Apdx. [A.5](#)) and not a direct value access but are still very fast. The extended connectivity hash codes were encoded in the range of ~ 0.1 ms per entry. The InChIKey accounts for bond orders (that were ignored in the hash code) and uniquely identifies more entries at the cost of a slower runtime.

2.3.4 Alignment of *iJR904* and MetaCyc 17.5

The use of hash codes had proved useful in matching metabolites between *iJR904* and MetaCyc. The improved hash code implementation resolved technical issues identified previously. However, the hash codes still only provided an all-or-nothing match. In particular, a structure can either match entirely with or without stereochemistry. To achieve a more precise match, a modified approach was adopted.

Candidate retrieval and categorisation

The metabolite entries in MetaCyc are loaded into a tree data structure for top-down retrieval (See. 2.2.2). Using an increasingly more selective hash code, a query metabolite (from *iJR904*) identifies one or more candidate matches in MetaCyc. If at some point no matches are present, the retrieval backs up and returns the best candidates at the previous level. The three levels of the tree are: formula-h, ec-4E, and ec-4ES. The choice of these three levels was based on the key distributions explored in the previous section. Bond orders are not encoded to allow matching of tautomers. After retrieval, candidates are standardised, scored, and eliminated (See. 2.2.3).

The term *candidate* is used to emphasise that a non-exact structure match may be identified. These matches may be correct but are associated with different levels of confidence (Tab. 2.6). Because the first retrieval level is formula-h, candidates are categorised as having the same element *composition* when no isomorphism is found.

Table 2.6: Category and confidence of retrieved candidates.

Type	Subtype	Confidence	Description
Identical	-	Very high	No observable difference
Stereoisomer	Less specific (single)	High	Unspecified stereocentre in <i>iJR904</i>
	Less specific	Medium	Unspecified stereocentres in <i>iJR904</i>
	More specific	Medium	Unspecified stereocentres in MetaCyc
	Unspecific	Medium	Unspecified stereocentres in <i>iJR904</i> and MetaCyc
	Diastereomers	Low	Inverted stereocentres (includes enantiomers)
Structurally related	Skeleton	Low	Heavy atoms, connected in the same way
	Composition	Remote	Same heavy atoms connect differently

iJR904 and MetaCyc 17.5 dataset differences

The *iJR904* reconstruction was imported and re-annotated with an updated version of Mergingear. Annotations were added from ChEBI using a name search; HMDB and

KEGG were also used. These were then manually inspected and corrected if needed. Additional annotation methods (described in the next chapter) were utilised when no structure was assigned through name searching. After annotation, there were 23 metabolites that had not been assigned structure representations (Apdx. A.7.1). Of those with structures, 125 were at a different protonation state to the annotated charge. The only entries with more than one structure annotation were 18 carbohydrates. These had been annotated with the chain (aldehydo) form and closed to the ring (furanose or pyranose) form using Metingear.

Release 17.5 of MetaCyc included corrections and updates to structural representations. Structures with stereochemistry indicated by projection in MetaCyc 16 had been redrawn, now allowing correct interpretation. MetaCyc provides structures as molfile, SMILES, and InChI. Although the methods described here can be applied equally on all three formats, the molfile representation was used from MetaCyc. The MetaCyc SMILES do not include stereochemistry and the InChIs have already been standardised.

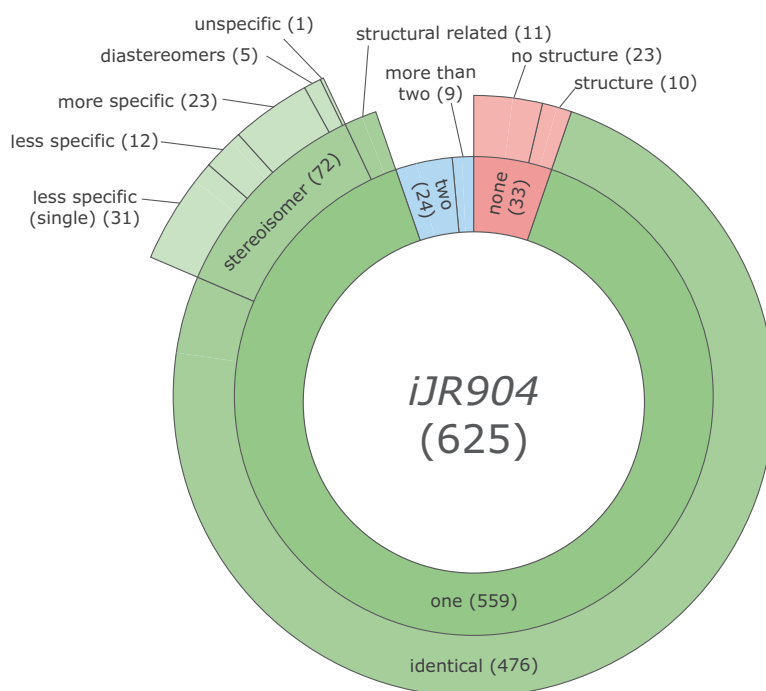
Retrieved candidates

Each metabolite in *iJR904* either retrieved: none¹, one, two, or more than two candidates from MetaCyc (Fig. 2.11a). Tables of retrieved candidates and expected matches are provided in Appendix A.7. The candidates that did not obtain a single identical match were systematically inspected.

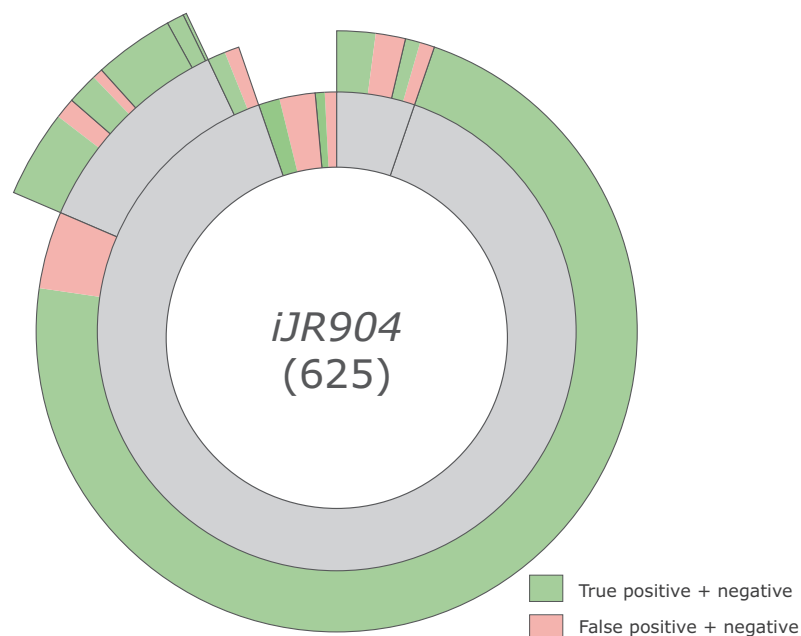
No candidates

A total of 33 metabolites did not identify any candidates in MetaCyc. Many of these were unspecific metabolites (part of the biomass and primarily required for simulation) and had not been annotated with a structure in *iJR904*. Those with structural annotations were fatty acids attached to the acyl-carrier-protein (ACP) and heme *o*. In MetaCyc the ACP-attached fatty acids do have structure representations but these are not provided in the MetaCyc release. The heme *o* MetaCyc entry uses non-standard bonds to represent coordination to the metal and was skipped during import.

¹In reality, when no match is found at the first level, all structures in the reference are returned. They are all candidates and a match may be obtained by other metrics such as names. For clarity these matches are referred to as none.



(a) Retrieved candidates



(b) Binary classification

Figure 2.11: Candidates in MetaCyc 17.5 retrieved for the 625 metabolites in *iJR904*. The inner circle indicates how many candidates were returned for each metabolite. These are then broken down into categories. The binary classification is compared to MNXref, false positives are counted when a candidate was either different or not listed.

One candidate

There were 559 *iJR904* metabolites that retrieved one candidate. The majority (476) of candidates were categorised as identical where no structural difference was observed.

The 72 stereoisomer candidates were inspected and found to be an appropriate match. Candidates identified as less specific occur when the *iJR904* entry has one or more stereocentres with no specified configuration. It may be that the metabolite is representative of multiple stereoisomers or that a single stereoform was intended. This information is not always explicitly annotated but it is perhaps inferred that a single form was intended when only one candidate is retrieved.

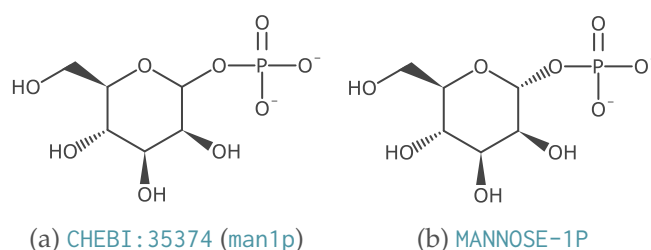


Figure 2.12: The *iJR904* metabolite D-mannose 1-phosphate ([man1p](#)) does not indicate whether the structure is the α or β anomer. The only candidate retrieved in MetaCyc was α -D-mannose 1-phosphate.

Most (31) of the 43 candidate categorised as less specific had a single unspecified tetrahedral stereocentre. Examining the structures identified 28 of the 31 candidates were a specific anomer (Fig. 2.12, Apdx. A.7.2).

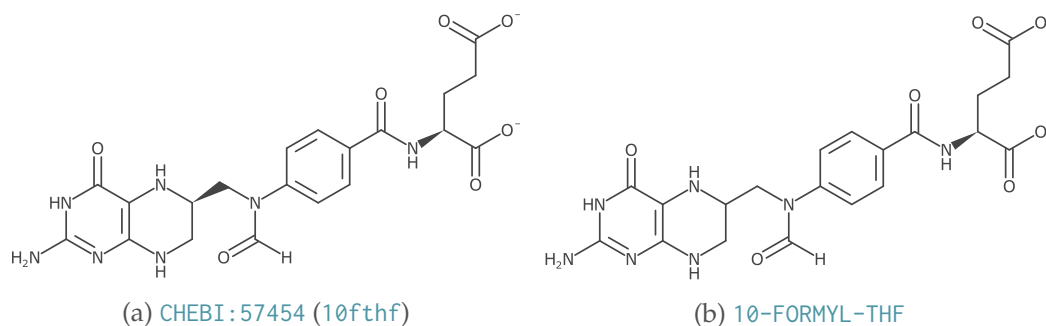


Figure 2.13: The metabolite 10-formyltetrahydrofolate (*10fthf*) was annotated to ChEBI in *iJR904*. A single candidate in MetaCyc was retrieved that had one unspecified stereocentre.

There were 23 candidates retrieved as more specific, these have matched an entry where one or more stereocentres in MetaCyc is undefined *iJR904* (Fig. 2.13, Apdx. A.7.2). Inspecting the MetaCyc entries found that 3 actually did have defined stereochemistry but it was either ambiguous or missed due to a deficiency in CDK stereo perception. There are 15 of the 20 that have updated and correct structures on the MetaCyc website, these do not appear to be distributed with the latest release.

The 5 diastereomers identified metabolites that were correctly matched but had a different configuration (Apdx. A.7.2). Manual inspection shows these matches were correct and either structure has been depicted with the incorrect stereochemistry. The MetaCyc website now depicts a corrected structure for 2 of these entries. The ChEBI and MetaCyc databases were notified of the difference in the other 3, CHEBI:28639 was found to represent the wrong configuration and has been corrected.

Candidates identified as structurally related have a low or remote chance of being a correct match. However, of the 11 structurally related candidates, 6 were found to be correct (Apdx. A.7.3). These are interesting cases as it means there was a difference in representation. They were all found to be tetrapyrrole derivatives such as porphyrin and corrin coordination complexes. Differences in proton locations resulted in only the skeleton matching (Apdx. A.6).

The other 5 structurally related entries were found to have missed the correct match in MetaCyc. These were not retrieved because the molfile of that (correct) entry was either not provided or due to a deficiency in the top-down retrieval. Testing stereochemistry without bond orders, a candidate with different saturation but identical stereochemistry is favoured over one with identical saturation but different or unspecified stereochemistry. Note, that this only happens when the correct match has

missing or wrong stereochemistry.

Two candidates

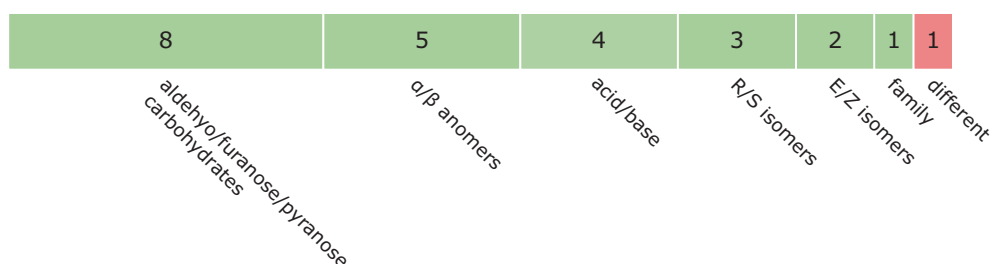


Figure 2.14: Classification of why two candidates in MetaCyc were found for a single *iJR904* metabolite. Matches are coloured as acceptable (green) or unacceptable (red) pairings. The single unacceptable pairing only matched as structurally related.

When a metabolite from *iJR904* identified two candidates it may be that it was representing two discrete entries. The most likely case for this is when two stereoisomers are identified for an entry with a single unspecified stereocentre. This was found to be the case for 10 entries (Fig. 2.14, Apdx. A.7.6.3, A.7.6.1).

There was 1 entry that had not found the best match. The metabolite butanoyl-CoA (*btcoa*) retrieved the candidates crotonyl-CoA and vinylacetyl-CoA as being structurally related. The butanoyl-CoA entry in MetaCyc had unspecified stereochemistry (in Coenzyme A) and was eliminated during retrieval. Since these crotonyl and vinylacetyl have the same skeleton and correct stereochemistry, they were selected instead.

The inclusion of ring and chain forms for carbohydrates in *iJR904* meant 2 entries (1 for each representation) were retrieved. This is fewer than those annotated in *iJR904* because only some carbohydrates in MetaCyc include representations for the different forms.

The neutralisation applied during retrieval removed distinctions between different protonation states. Conjugate acid and bases are then seen as identical. Finally, a single entry was found where a parent and child entry in MetaCyc used the same structure.

More than two candidates

From *ijR904*, 9 entries were found to match to more than two candidates (Apdx. A.7.6.4, A.7.6.2). There were 7 appropriately matched entries that were: stereoisomers (2), at different protonation states (4), or at different oxidation states (1). The stereoisomers here have more than one stereocentre without a configuration. These identified more than 2 candidates with fully specified stereocentres in MetaCyc.

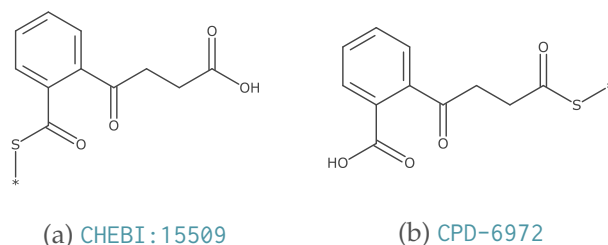


Figure 2.15: The *ijR904* metabolite o-succinylbenzoyl (*sbzcoa*) was annotated to CHEBI:15509. The correct entry in MetaCyc (CPD-6972) was attached to coenzyme A via the other carboxylate.

There were 2 metabolites that retrieved correct entries but only as structurally related.

The first entry was due to a different attachment point between coenzyme A and multiple carboxylate groups (Fig. 2.15). The structure representations are correct for the primary name: 2-succinylbenzoyl-CoA (ChEBI) and 4-(2'-carboxyphenyl)-4-oxobutyryl-CoA (MetaCyc). Both are also annotated with the synonym o-succinylbenzoyl. On inspection by the ChEBI curators, the enzyme (EC 6.2.1.26) is listed as producing 2-succinylbenzoyl-CoA. However, 4-(2'-carboxyphenyl)-4-oxobutyryl-CoA is the substrate in a subsequent reaction (EC 4.1.3.36). This appears to be a misnomer in the enzymatic reaction EC 6.2.1.26. The ChEBI entry has been updated and this information passed on to other databases (UniProt and ENZYME).

The second entry that found structurally related candidates was an unsaturated fatty acid. Since the location of the double bond is not specified, it may be located at different locations along the alkyl-chain. No entry was found as an identical match and so 22 fatty acids with the same heavy element composition were returned as candidates.

Table 2.7: Binary classification of retrieved candidates compared to MNXref. The false positive classification includes both when a different match was found or a match was found when no match was listed (unmapped). The number of unmapped false positives is shown in parentheses.

Candidates	Type	Subtype	TP	FP	TN	FN	Precision (cumulative)	Sensitivity (cumulative)
0		no structure	0	0	13	10	-	0.000
0		with structure	0	0	5	5	-	0.000
1	identical		451	25 (13)	0	0	0.947	0.968
1	stereoisomer	less specific (single)	25	6 (5)	0	0	0.939	0.969
1	stereoisomer	less specific	9	3 (3)	0	0	0.934	0.970
1	stereoisomer	more specific	23	0	0	0	0.937	0.971
1	stereoisomer	unspecific	1	0	0	0	0.937	0.971
1	stereoisomer	diastereomer	5	0	0	0	0.938	0.972
1	structurally related	skeleton	6	4 (2)	0	0	0.932	0.972
1	structurally related	composition	0	1	0	0	0.930	0.972
2			13	11 (1)	0	0	0.914	0.973
>2			6	3 (3)	0	0	0.910	0.973

Binary classification

The MNXref database was used to evaluate the candidate retrieval. The database includes reconciled entries from both the *iJR904* (from BiGG) and MetaCyc 17.5. Retrieved candidates were classified as true or false positives if the entry was different from that listed in MNXref. It is important to note that the false positive count includes when an metabolite in *iJR904* has no mapping assigned by MNXref but a candidate was retrieved using the hash codes. True and false negatives were counted when no candidate was retrieved and whether this was expected. The distribution of classes across the categories is displayed in Figure 2.11b and listed in Table 2.7.

As MNXref uses additional details such as names, 15 false negatives without structure representations in *iJR904* or MetaCyc were listed in MNXref (Apdx. A.7.4).

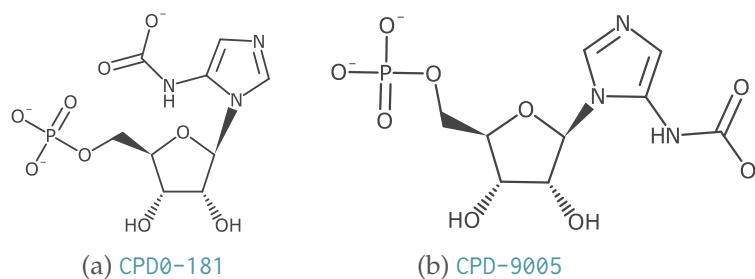


Figure 2.16: The metabolite 5-phosphoribosyl-5-carboxyaminoimidazole (*5caiz*) was a false positive and matched to different entries in MetaCyc. The MetaCyc entry CPD0-181 was found using the structure based retrieval and CPD-9005 by MNXref. These are the same metabolite and appear to be a duplicate in MetaCyc.

Compared to MNXref, there were 25 false positives that were categorised as identical candidates. Of these 25, 12 mapped to a different MetaCyc entry and 13 had no mapping assigned in MNXref (Apdx. A.7.5.1). Half of the 12 different entries were equivalent, either being duplicates (Fig. 2.16) or a parent and child compound class. In MNXref, L-fucose is correctly mapped to the L-fucose (no structure) compound class, but the candidate L-fucosepyranose (a child class) was retrieved based on structure.

Another 3 candidates were unspecified anomers, the metabolite UDPglucose [sic] ([udpg](#)) retrieved the candidate UDP-D-glucose ([UDP-GLUCOSE](#)) but MNXref had mapped to UDP- α -D-glucose ([CPD-12575](#)).

The remaining 3 that mapped to different entries were seen to be due to incorrect annotation of *ijR904*. Here, incorrect structures were annotated due to ambiguous or incorrect naming. The metabolite, dihydroneopterin monophosphate ([dhmp](#)) does not state the attachment point of the phosphate group. Therefore, the annotated structure had the phosphate attached in one location but the expected MetaCyc entry (by MNXref) had the phosphate in a different location. Similarly, undecaprenyl phosphate ([udcpp](#)) has ten geometric stereocentres. The structure that had been annotated was a different stereoisomer from that mapped by MNXref. The systematic name for the lipid A metabolite was incorrect in *ijR904* (from the original publication) and so the wrong structure was assigned. MNXref was created with a more recent version (from BiGG) in which this name has been updated.

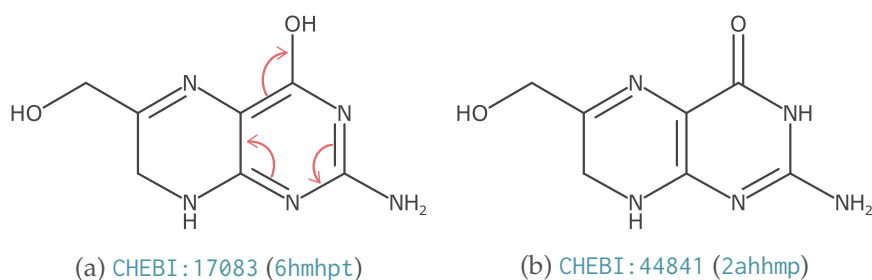


Figure 2.17: The BiGG database contains 6-hydroxymethyl dihydropterin ([6hnhpt](#)) and 2-amino-4-hydroxy-6-hydroxymethyl-7,8-dihydropteridine ([2ahhmp](#)). These are annotated as different but never occur in the same reconstruction. The ChEBI entries for these two metabolites are annotated as tautomers.

The 13 identical candidates that were not listed by MNXref were inspected and found to be appropriate matches (Apdx. A.7.5.1). There were 2 interesting entries that appeared to be tautomers. One of these was 6-hydroxymethyl dihydropterin ([6hnhpt](#))

in *iJR904*, which had identified 6-hydroxymethyl-7,8-dihydropterin¹ as a candidate in MetaCyc. MNXref did not list a mapping for [6hnhpt](#) but did map the MetaCyc entry to a different BiGG entry ([2ahhmp](#)). ChEBI includes entries for both structures and they are annotated as tautomers (Fig. [2.17](#)).

All 9 false positives that retrieved a stereoisomer candidate were categorised as having less specific stereochemistry. Only 1 of these entries was mapped to a different MetaCyc entry by MNXref. Through the structure retrieval, the metabolite dTDP-4-amino-4,6-dideoxy-D-glucose identified the α anomer dTDP-4-amino-4,6-dideoxy- α -D-glucose ([CPD-472](#)) as a candidate. The MNXref mapping was to dTDP-4-amino-4,6-dideoxy-D-galactose ([CPD-14020](#)) that has inverted stereochemistry. The 8 candidates not listed by MNXref were seen to be an appropriate match (Apdx. [A.7.5.2](#)).

The 5 (2 unmapped) false positives categorised as structurally related were incorrectly mapped due to reasons explored earlier. The matches were not found because the correct MetaCyc entry either did not provide the structure or had incorrect stereochemistry leading to a different candidate being retrieved (Apdx. [A.7.5.3](#)).

The false positives that identified more than one candidate were due to mainly matching the children instead of the parent compound class (Apdx. [A.7.6.1](#), [A.7.6.2](#)). The only incorrect match was for butanoyl-CoA, which was discussed previously.

There were two cases where MNXref did not list a mapping but stereoisomers were found: isocitrate ([icit](#)) and D-mannose 6-phosphate ([man6p](#)). It is interesting to note the mapping of *o*-succinylbenzoyl-CoA was not listed by MNXref.

¹AMINO-OH-HYDROXYMETHYL-DIHYDROPTERIDINE

2.3.5 Alignment of *iYO844* and *iBsu1103*

The analysis of *iJR904* and MetaCyc 17.5 demonstrated the ability to align metabolites from a reconstruction to a database. To explore how metabolites could be aligned between reconstructions, two models of *Bacillus subtilis* were utilised.

The *iYO844*^[168] and *iBsu1103*^[143] reconstructions were chosen for two reasons. Firstly, the *iBsu1103* model was published with chemical structure annotations (molfile). Secondly, as the same organism, *iBsu1103* includes a curated mapping to *iYO844* that allowed for validation.

The *iYO844* and *iBsu1103* reconstructions were imported into Metingear. The molfiles for *iBsu1103* were attached to the reconstruction unmodified. Metingear was used to annotate *iYO844* with structure representations. The primary resource used in annotating *iYO844* was ChEBI. Automated mappings found by name matches underwent some manual curation where needed. Structures for dipeptides and fatty acids were manually added. Annotation to KEGG was avoided to ensure the representations in *iBsu1103* were not assigned. Structural annotations were not included for 43 un-specific metabolites (Apdx. A.8.1). As with *iJR904*, 228 structures were at a different protonation state to that annotated.

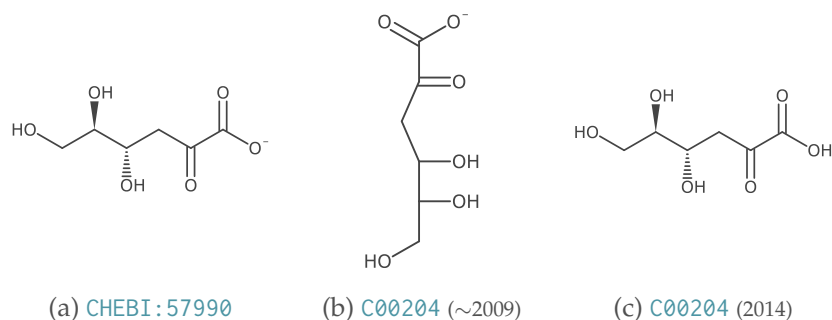
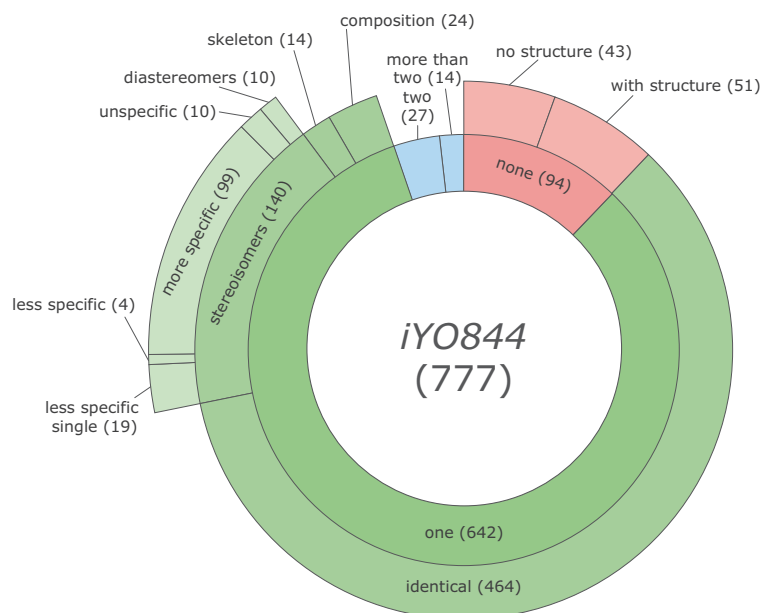


Figure 2.19: Chemical structure representations of 2-dehydro-3-deoxy-D-gluconate (2ddg1cn). Previous releases of the KEGG database only indicate stereochemistry via a perspective projection.

The top-down retrieval was used to identify candidates in *iBsu1103* that matched metabolites in *iYO844*. The parameters for retrieval were the same as for *iJR904* and MetaCyc 17.5, the categories are also the same (Tab. 2.6).

A total of 642 of the 777 metabolites in *iYO844* found a single candidate match in *iBsu1103* (Fig. 2.18a). A large portion (140) of these candidates were stereoisomers of



(a) Retrieved candidates



(b) Binary classification

Figure 2.18: Overview of candidates retrieved in *iBsu1103* for metabolites in *iYO844*. The inner circle indicates how many candidates were identified for each entry. The subtypes of each candidate are then displayed. The binary classification is compared to a mapping annotated in *iBsu1103*, false positives were counted when a different or unmapped candidate was retrieved.

which most had more specific stereochemistry. Inspection of the entries in *iBsu1103* identified that many structures annotated from KEGG only indicated stereochemistry by projection (Fig. 2.19). These configurations cannot be recognised and are treated as though stereochemistry is missing.

There were 51 metabolites annotated with chemical structures from *iYO844* that did not identify any candidates in *iBsu1103*. This is surprising as the reconstructions are of the same metabolome. *iBsu1103* was created after *iYO844* and does not include any metabolites that resemble these entries.

Binary classification

The mapping provided in *iBsu1103* was used to classify retrieved candidates. When more than one candidate was retrieved a true positive was counted if any candidate was correct (Fig. 2.18b). The majority of false positives were candidates that were not listed in *iBsu1103* rather than being different mappings (Tab. 2.8). Supplementary tables listing retrieved candidates and expected matches are provided in Appendix A.8.

Table 2.8: Binary classification of matched candidates across different categories. The precision and sensitivity is indicated as a cumulative sum of classifications descending in confidence. The false positives include both those that are different from that indicated in *iBsu1103* and those that were not listed (unmapped). The number of unmapped false positives is shown in parentheses.

Candidates	Isomer type	Subtype	TP	FP	TN	FN	Precision (cumulative)	Sensitivity (cumulative)
0			0	0	30	13	-	0
0			0	0	46	5	-	0
1	identical		461	3 (2)	0	0	0.993	0.962
1	stereoisomer	less specific (single)	17	2 (2)	0	0	0.989	0.963
1	stereoisomer	less specific	4	0	0	0	0.989	0.964
1	stereoisomer	more specific	93	6 (6)	0	0	0.981	0.969
1	stereoisomer	unspecific	9	1	0	0	0.979	0.970
1	stereoisomer	diastereomer	5	3 (3)	0	0	0.975	0.970
1	structurally related	skeleton	6	8 (6)	0	0	0.962	0.970
1	structurally related	composition	10	14 (14)	0	0	0.942	0.971
2			16	11 (10)	0	0	0.928	0.971
> 2			6	8 (7)	0	0	0.918	0.972

False negatives were found for entries both with and without structure annotations. Entries mapped without structure annotation included proteins, polysaccharides, and

tRNA attached metabolites. Only 5 metabolites annotated with structures did not match due to 1 salt and 4 tetrapyrroles (Apdx. A.8.2).

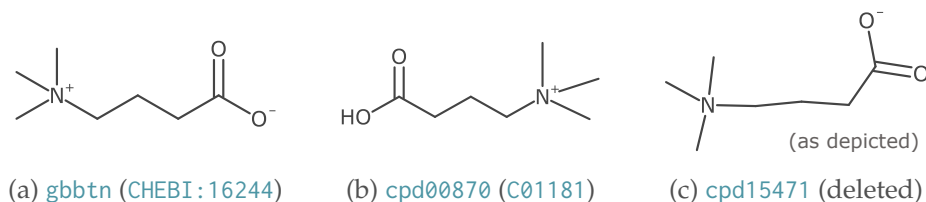


Figure 2.20: Structure representations of gamma-butyrobetaine (*gbbtn*), 4-trimethylammonio-butanoate (*cpd00870*), and gamma-butyrobetaine (*cpd15471*). The *cpd00870* and *cpd15471* entries from *iBsu1103* were identified as duplicate.

Only 3 false positives were identical candidates (Apdx. A.8.3.1). The *iYO844* metabolite gamma-butyrobetaine (*gbbtn*) found the candidate match of 4-trimethylammonio-butanoate (*cpd00870*). The expected mapping in *iBsu1103* was to gamma-butyrobetaine (*cpd15471*). Inspection of the structure representation identified the metabolites were duplicates in *iBsu1103* (Fig. 2.20). The *cpd15471* entry no longer exists in the most recent version of the SEED database. The different names are listed as synonyms in both ChEBI and KEGG. Only one candidate was retrieved because *cpd15471* was missing a charge on the nitrogen, the elimination removed this candidate as scoring worse than *cpd00870*.

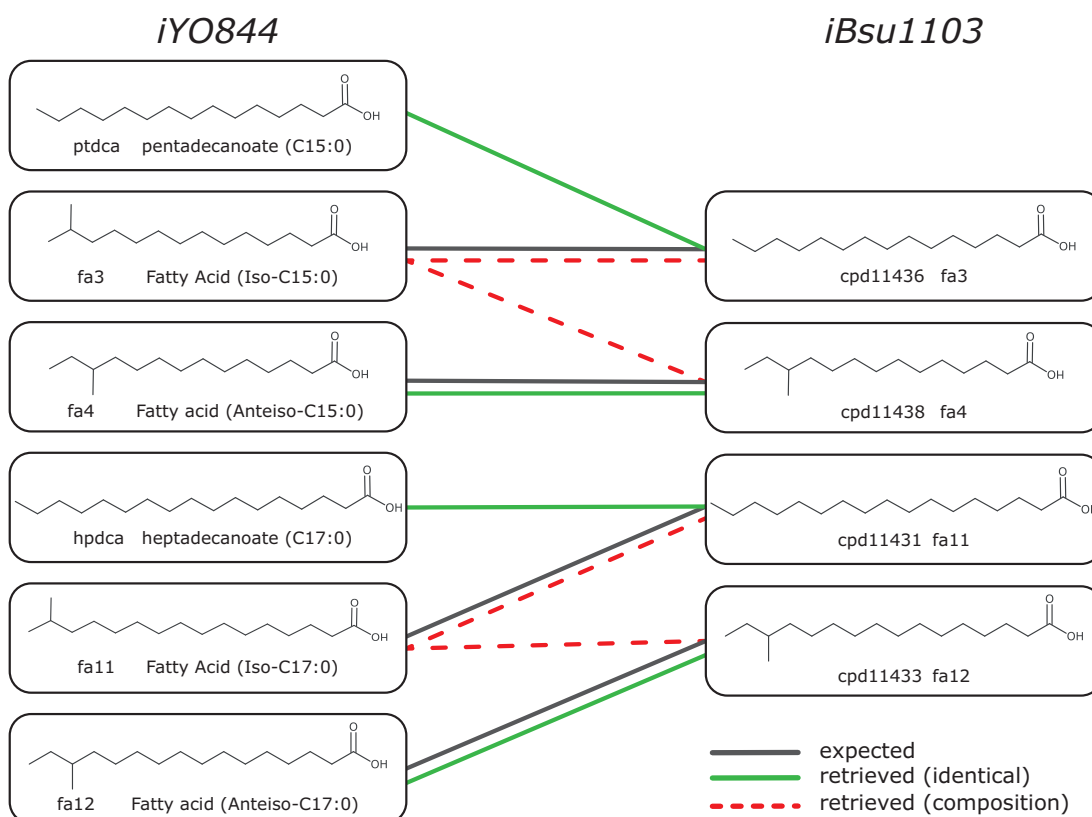


Figure 2.21: Unexpected fatty acid matches in *iBsu1103* were retrieved as the depictions of fa3 and fa11 did not match that assigned to *iYO844*. The synonyms of fa3 and fa11 in *iBsu1103* indicate the alkyl chain has an iso arrangement, this is not represented in the structure annotations.

The other 2 false positives were candidates that were not listed in the *iBsu1103* mapping. The two entries pentadecanoate (ptdca) and heptadecanoate (hpdca) appeared to match the metabolites fa3 (cpd11436) and fa11 (cpd11431) in *iBsu1103*. This mapping was not listed, but fa3 and fa11 instead mapped to two other fatty acids (of the same abbreviation) in *iYO844*. Expanding out the candidates and mappings between the reconstructions reveals an inconsistency (Fig. 2.21).

Based on the information available in *iYO844* the annotation of fatty acid structures is consistent with the iso- and anteiso- prefixes. At the time of writing, the entries in the SEED database are also annotated with the same representation in *iBsu1103*.

There were 12 false positives categorised as stereoisomers (Apdx. A.8.3.2). Only 1 of these was a false positive that matched a different entry. The metabolite hexadecenoyl-

CoA ([hdcoa](#)) has an unspecified double bond location, it matched trans-hexadec-2-enoyl-CoA ([cpd03126](#)) in *iBsu1103* because the unspecified location defaults to the second atom in the chain. The expected mapping had the unspecified double bond in a different location and was not retrieved.

Some of the stereoisomer candidates retrieved when no mapping was expected had also be retrieved as identical for another metabolite. The *iYO844* reconstruction contains both ala-ala ([diala-L](#)) and D-alanyl-D-alanine ([alaala](#)) but *iBsu1103* only contains D-alanyl-D-alanine ([cpd00731](#)). The [diala-L](#) metabolite was categorised as a stereoisomer to [cpd00731](#) but [alaala](#) was scored as identical. By inspecting this, the mapping from [diala-L](#) could be automatically eliminated.

The metabolites maltohexaose ([malthx](#)) and maltopentaose ([maltpt](#)) happened to have the same representation as the dextrin ([cpd11594](#)) and amylose ([cpd11735](#)) polysaccharides. Here, the polysaccharides are misrepresented as a single non-repeating unit.

The candidates retrieved for 4 fatty acids appear to be correct but were not listed in the *iBsu1103* mapping:

<i>iYO844</i>		<i>iBsu1103</i>	
Abbreviation	Name	Abbreviation	Name
fa12coa	Anteiso-C17:0 CoA	cpd11434	fa12coa
fa1coa	Iso-C14:0 CoA	cpd11435	fa1coa
fa3coa	Iso-C15:0 CoA	cpd11437	fa3coa
fa4coa	Anteiso-C15:0 CoA	cpd11439	fa4coa

These were not categorised as identical as coenzyme A was represented using a projection in *iBsu1103*. The naming scheme appears to indicate these metabolites were correctly matched. Note that compared to fa3 ([cpd11436](#)) (discussed previously), fa3coa ([cpd11437](#)) structure does depict the iso arrangement of the alkyl chain.

Another potential match was a diastereomer, D-*myo*-inositol 1,3,4,5,6-pentakisphosphate ([inospp1](#)) retrieved the *iBsu1103* candidate D-*myo*-inositol 1,2,4,5,6-pentakisphosphate ([cpd02780](#)). The difference between these is a phosphate has been removed from a different hydroxy group. No expected mapping was indicated for each metabolite but both occur in a single reaction and are the product of cleaving a phosphate from *myo*-inositol hexakisphosphate ([inoshp](#)). Both of the reactions are labelled as [EC 3.1.3.8](#) and the *iBsu1103* metabolite appears to be a misnomer. Indeed, the reaction in *iBsu1103* is the only one listed in SEED whilst D-*myo*-inositol 1,3,4,5,6-penta-

kisphosphate ([cpd00943](#)) participates in eight reactions.

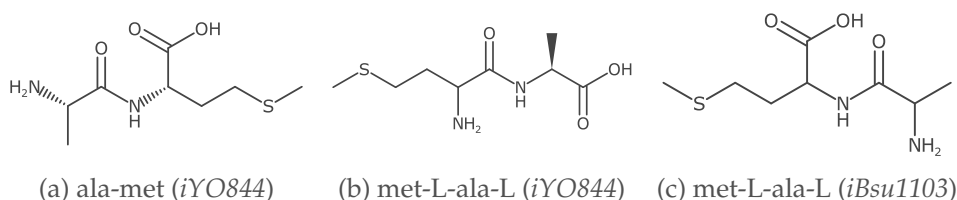
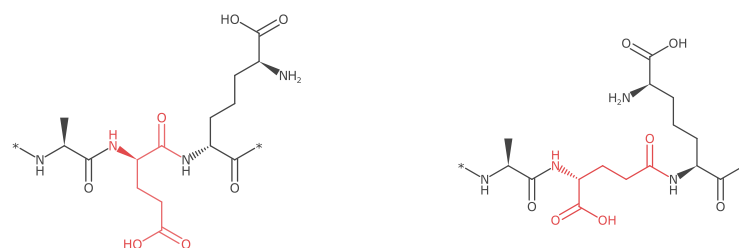


Figure 2.22: In *iBsu1104* met-L-ala-L ([2.22c](#)) is depicted in reverse with the alanine residue at the N- terminus. It was therefore identified as a stereoisomer (missing stereochemistry) of ala-met ([2.22a](#)) but as only structurally related to met-L-ala-L ([2.22b](#)).

An interesting case was seen for the dipeptide ala-met ([ala-L-met-L](#)) that had identified the stereoisomer candidate met-L-ala-L ([cpd11590](#)). This match was identified because the structure representation for met-L-ala-L in (*iBsu1103*) was reversed from C- terminus to N-terminus (Fig. [2.22](#)). This was not the case for all dipeptides as ala-L-asp-L ([cpd11593](#)) is represented from N- to C- terminus in *iBsu1103* and SEED.

The high number of false positive candidates categorised as structurally related was expected as this is a low confidence assignment. Only 2 false positives had found a different mapping (Apdx. [A.8.3.3](#)). Similar to *iJR904*, the choice of keys for the structural retrieval meant entries with missing stereochemistry were filtered out during retrieval. Here the expected match has missing stereochemistry represented as a projection, whilst the candidate had specified stereochemistry. The other false positives were not previously listed but did not indicate any missed mappings.

The structurally related true positives are of interest because the structures were different. There were 16 true positives that only matched based on either having the same skeleton or heavy atom composition. Of these 16, 7 were tetrapyrrole derivatives with different saturation or substituent locations (Apdx. [A.6](#)) and 4 entries that had not be standardised properly (e.g. unspecified protonation). Another 3 were again due to reversed dipeptides and had only matched with heavy element composition (Apdx. [A.8.3.4](#)).



(a) -D-glutamyl- (C05898, cpd03495) (b) -D-γ-glutamyl- (CHEBI : 28138)

Figure 2.23: Part of the metabolite undecaprenyl-diphospho-*N*-acetylmuramoyl-(*N*-acetylglucosamine)-*L*-alanyl-D-glutamyl-meso-2,6-diaminopimeloyl-D-alanyl-D-alanine (uaagmda), the D-glutamyl may be bonded via the backbone or sidechain (γ).

There were 2 true positives due to structural differences between ChEBI and KEGG. One of these was the *o*-succinylbenzoyl attachment to coenzyme A (Fig. 2.15) discussed earlier. The other was due to a γ-attachment of an oligopeptide group (Fig. 2.23). The KEGG entry (C05898) has since been modified and now has the same representation as ChEBI, SEED has not be updated.



(a) crotonobetaine (ctbt) (b) crotono-betaine (cpd08305)

Figure 2.24: The structure annotated crotono-betaine in *iBsu1103* is incorrect. The KEGG entry (C11458) which was used in *iBsu1103* no longer exists.

Of entries that matched two candidates, only 1 false positive had an existing entry mapped to *iBsu1103* (Apdx. A.8.4.1). The entry crotonobetaine (ctbt) did not retrieve the expected crotono-betaine (cpd08305) due to different structures (Fig. 2.24). The structure annotated in *iBsu1103* was retrieved from a now deleted KEGG entry (C11458). The SEED entry still contains this representation. The other false positives were unmapped in *iBsu1103* and matched only as structurally related candidates, no new mappings were identified.

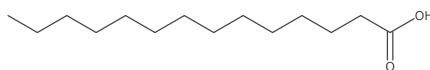


Figure 2.25: The structure of tetradecanoate ([tttdca](#)) is annotated for both tetradecanoic acid ([cpd03847](#)) and fa1 ([cpd11430](#)) in *iBsu1103*. As with fa3 and fa11, fa1 should be represented as iso-.

The true positives that mapped to two candidates were due to tetrapyrroles or duplicate structures for fatty acids (Fig. 2.25, Apdx. A.8.4.3).

There were 14 metabolites in *iYO844* that retrieved more than two candidates in MetaCyc. Of the 8 false positives only 1 had an existing mapping in *iBsu1103*. Surprisingly, this entry was for the proton ([h](#)). The H^+ ([cpd00067](#)) in *iBsu1103* structure used a pseudo atom annotated with the label – “H\S+\n”, this has been modified in the most recent release of SEED. This entry retrieved one-to-many as 4 structures in *iBsu1103* were not loaded and produced the same hash code as the proton (i.e. nil). The other false positives that were unmapped were only structurally related (Apdx. A.8.4.2).

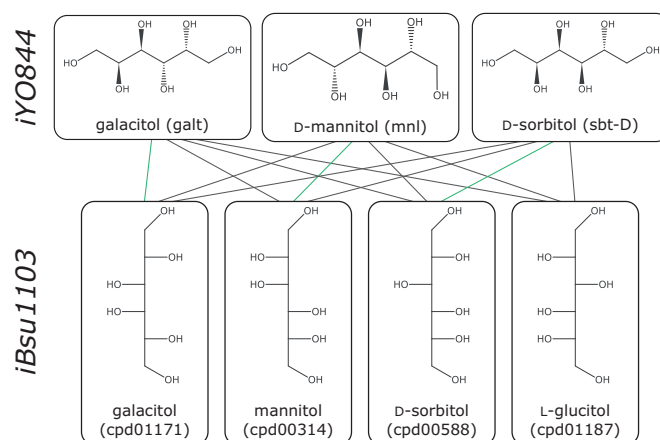


Figure 2.26: Stereochemistry indicated by Fischer projection in *iBsu1103* is not recognised and sugar alcohols in *iYO844* all identified the same four candidates.

The true positives did not resolve to a single entry due to missing stereochemistry in *iBsu1103* (Fig. 2.26), unspecified double bond placement and the reverse dipeptide representations (Apdx. A.8.4.4).

2.4 Conclusions

This chapter has introduced procedures to find equivalent metabolites based only on chemical structure. Reconstructions (and a database) are aligned on-demand using only the content of the reconstructions. To provide a fast and flexible identity check, connectivity hash codes are used. The implementation is openly available in the Chemistry Development Kit allowing modification and extension as needed. The scoring and candidate retrieval is available via MDK and currently being integrated into Metingear (Chap. 3).

The exploratory analysis of *iJR904* identified several limitations. This analysis drove the improvement of the hash code and development of algorithms that are described in this and subsequent chapters.

Investigating collisions in the ChEBI database identified duplicates and inconsistencies. As a consequence of this analysis, these issues were addressed and the quality of the database has improved. The analysis of PubChem-Compound identified a deficiency in the original algorithm to discriminate non-equivalent atom environments. By increasing the level of perturbation, every entry in PubChem-Compound could be assigned a distinct hash code.

Compared to the InChIKey the hash code discriminated PubChem-Compound entries that may be seen as equivalent. Key areas to address are related to standardisation of: salts, mobile charges, and ignoring geometric isomerism of tautomeric bonds. It should be noted that these were distinct entries in PubChem-Compound and it is therefore useful to be able to distinguish them.

By scoring retrieved candidates with a full isomorphism check, algorithm correctness (no collisions) is guaranteed and worse scoring candidates can be eliminated. The scores are used to refine matched candidates and automatically categorise whether the structures were identical, stereoisomers (and their type), or structurally related.

Using a top-down retrieval with increasingly more sensitive encodings allows for rapid identification. When no match is identified, the retrieval backs up and returns the best candidates that could be found. With the choice of keys used in this analysis, a candidate with different saturation may be selected over one with identical saturation. This happens when the expected match has missing or inverted stereochemistry, which was found for several entries.

Being able to back up more than one level, the correct candidate could be found. To encode bond orders before stereochemistry is another option but would mean that different tautomers obtain distinct hash codes.

Candidates are standardised after retrieval but before scoring. Performing this optionally after scoring would allow candidates to be additionally categorised as acids, bases, and tautomers. The neutralisation step also resulted in multiple candidates being scored as equivalent (e.g. NH_3 and $[\text{NH}_4]^+$). In these cases, the candidate with the matching protonation could be chosen by selectively applying neutralisation only when no identical match is found. These may still need to be merged depending on the reactions they are involved in.

Re-evaluating the method on reconstructions identified duplicates and inconsistencies in structures from: ChEBI, KEGG, MetaCyc, and SEED. Some of these issues have subsequently been resolved, although this demonstrates both a limitation and advantage. Identified candidates are only as good as the annotations present, an incorrect structure annotation then directly leads to an incorrect mapping. However, this emphasises the utility in efficiently matching entries on-demand and an improved annotation automatically improves the mapping. Unambiguously representing all metabolite entries remains a challenge and shall be discussed in the main conclusions.

The candidate retrieval of *iJR904* to MetaCyc was compared to the MNXref database and the candidates systematically inspected. The automated procedure used by MNXref exploits additional information such as names and reaction context. Using nothing but the structure representations, the majority of expected mappings listed by MNXref were retrieved.

When considering only the false positives that had a different mapping, the incorrect matches were due to matching the child compound class (e.g. D-glucopyranose instead of D-glucose) or a stereoisomer. In total there were only 10 retrieved candidates that had a more appropriate mapping. Of these, 3 were due to naming inconsistencies and 7 were only structurally related. As the structurally related candidates are low confidence, they would warrant manual inspection. Compared to those annotated in MNXref, 17 additional mappings were found. Many of these were stereoisomers but would constitute an appropriate alignment.

The candidate retrieval was also evaluated on aligning the *iYO844* and *iBsu1103* reconstructions. Duplicates were identified and inconsistencies were seen in fatty acid and dipeptide structure representations of *iBsu1103*. These were responsible for most

of the false positives. The other false positives primarily only matched as structurally related. In addition to the existing mappings, 5 new mappings were found.

The alignment is currently being integrated in Metingear. This will allow efficient systematic inspection of non-exact matches and allow mappings for these to be assigned with user confirmation.

A limitation of the technique is when a metabolite does not have a structural annotation. By extending the approach to inspect reactions and names the alignment could be improved. However, using names may actually provide a less useful match. Using the name of metabolites, MNXref correctly identifies the metabolite ornithine (*orn*) as matching the MetaCyc compound class ornithine (*25-DIAMINOPENTANOATE*). As this entry does not have a structure representation (via download) the enantiomers were retrieved in this analysis (Fig. 2.27). MetaCyc only directly lists reactions for the enantiomers and not the parent compound class so the enantiomers in this case are more useful.

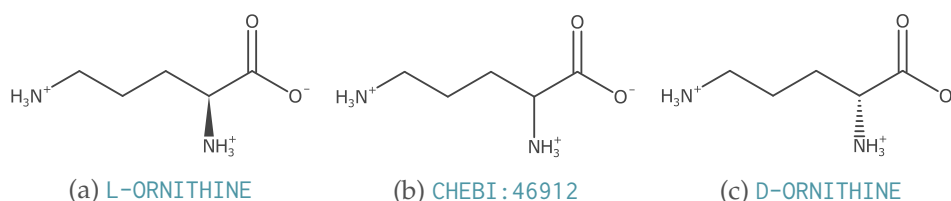


Figure 2.27: The *iJR904* metabolite ornithine (*orn*) (2.27b) retrieved L-ornithine (2.27a) and D-ornithine (2.27a) as stereoisomer candidates in MetaCyc.

An important application for aligning reconstructions is resolving network gaps. Resolving these gaps may require adding reactions from an external reference (e.g. MetaCyc). If a mapping is to a metabolite that is neither consumed or produced in the reference, no reactions involving that metabolite can be utilised.

2.5 Availability

The hash code and isomorphism mapping algorithms are available in the CDK source code repository: <http://www.github.com/cdk/cdk>. The relevant Java class packages and names are:

- `org.openscience.cdk.hash.*` – hash code package
- `org.openscience.cdk.hash.HashGeneratorMaker` – fluent API for configuring hash codes

-
- [org.openscience.cdk.isomorphism.VentoFoggia](https://github.com/orgs/openscience.cdk/projects/10) – backtracking isomorphism matching

The annotated *iJR904*, *iYO844*, and *iBsu1103* reconstructions utilised in this chapter are available at <http://johnmay.github.io/metingear/dissertation/>.

The retrieval and categorisation is being integrated in Metingear. The source code is available in the MDK search-tree module.

- <https://github.com/johnmay/mdk/tree/develop-1.5/tool/search-tree>
- <http://www.github.com/johnmay/metingear>

Chapter 3

Annotation of reconstructions

3.1 Introduction

3.1.1 Motivation

The increased availability of genome-scale metabolic reconstructions requires their content is interpretable and can be reused. Traditionally, non-standard tabular formats were the de facto standard for distributing experimentally verified reconstructions. The onset of standard formats, such as SBML, has improved the exchange of models through well defined specifications and software libraries^[169,24].

Although standard formats describe how the components of a reconstruction interact, what the actual components are may still be ambiguous. By annotating components with external resource identifiers, additional information may be retrieved and used. The MIRIAM registry provides a mechanism to unambiguously encode resource links^[27] that are now frequently included. Identifiers encoded in this manner are database dependent and additional effort is required to determine if two components are equivalent. As presented in Chapter 2, annotation of metabolites with a chemical structure representation provides a database independent identifier that can also be used to align, improve, and analyse reconstructions.

The annotation of database identifiers and chemical structures should be added during the refinement stage of construction^[11]. This stage involves extensive manual curation but the scale of reconstructions means it is impractical to edit the file content

directly. Doing so may introduce missing components or make the format uninterpretable. Tools that manage the creation, modification, and annotation of reconstructions are therefore needed.

3.1.2 Existing software

Existing software solutions vary in their support for standard formats, how reconstructions are represented, and how annotations are included, used, and stored. The representation and annotation aspects of tools that have similar functionality to that discussed in this chapter are summarised below.

The Pathway Tools^[56] software allows manual annotation of metabolites with multiple cross-references and a chemical structure annotation. Organism specific reconstructions can inherit these annotations from the central MetaCyc database. Annotations are stored in the internal format but are not currently exported to SBML. Chemical structure annotations can be searched and are used to create atom-atom mappings^[123] of reactions and find metabolic routes^[131].

The SuBliMinaL Toolbox^[61] represents and modifies reconstructions using the standard JSBML library^[169]; input from KEGG and MetaCyc is converted to SBML. Annotation is provided by libAnnoationSBML^[62] that can automatically add annotations to multiple resources using a name search. Annotations can also be selected from a list of suggestions using a name search. All annotations are semantically encoded using MIRIAM resource identifiers. Using libAnnoationSBML, chemical structures can be retrieved from an annotation and these are then passed to JChem^[170] to determine the protonation of a metabolite at physiological pH.

The RAVEN^[66] toolbox allows the inclusion of multiple cross-references and an InChI string. Reconstructions can be imported and exported from model formats including Excel and SBML. The Excel import requires a standardised column naming. SBML is generated without a standard library and MIRIAM resource annotations are not correctly encoded (Apdx. B.1).

semanticSBML is a desktop and web application for the annotation, validation and merging of SBML models^[171]. SBML entities are annotated with MIRIAM resource annotations. Annotations can be input manually or found with a string search from one or more data resources. Similar to semanticSBML, SAINT is a web application dedicated to the annotation of SBML and CellML models^[172]. Annotation in SAINT

is focused on the protein and reaction entries rather than metabolites.

GEMSiRV^[76] can import reconstructions from a table, provided the required columns are in the correct order. Annotations are fixed for each metabolic entity only allowing cross referencing to KEGG and CAS¹. Annotations are neither automated nor exported to SBML. Metannogen^[173] allows manual editing of annotations of SBML files locally and via a group annotation server; no assistance is provided for creating new annotations.

3.1.3 Objectives

To effectively curate, modify, and manage metabolic reconstructions an interactive application is required. The interaction allows one to verify annotations are correct and modify them if required. The annotation needs to be flexible and not restricted to discrete predefined values. Using resolvable identifiers from the MIRIAM registry provides this for cross-references.

It is important that data is obtainable from primary databases rather than an aggregated collection. This allows any updates to these resources to be utilised and avoids error propagation discovered in derived resources (Chap. 2).

The chemical structure of metabolites is an important annotation and should be treated as a first-class field. To achieve this, integration with the Chemistry Development Kit is required for displaying and modifying structures.

Towards these objectives, the open source Model Development Kit (MDK) library and Metingear desktop application were created. This chapter describes the implementation and functionality of both the library and application. The assistance Metingear provides for annotating reconstructions is explored.

¹The Chemical Abstract Service, <http://www.cas.org>

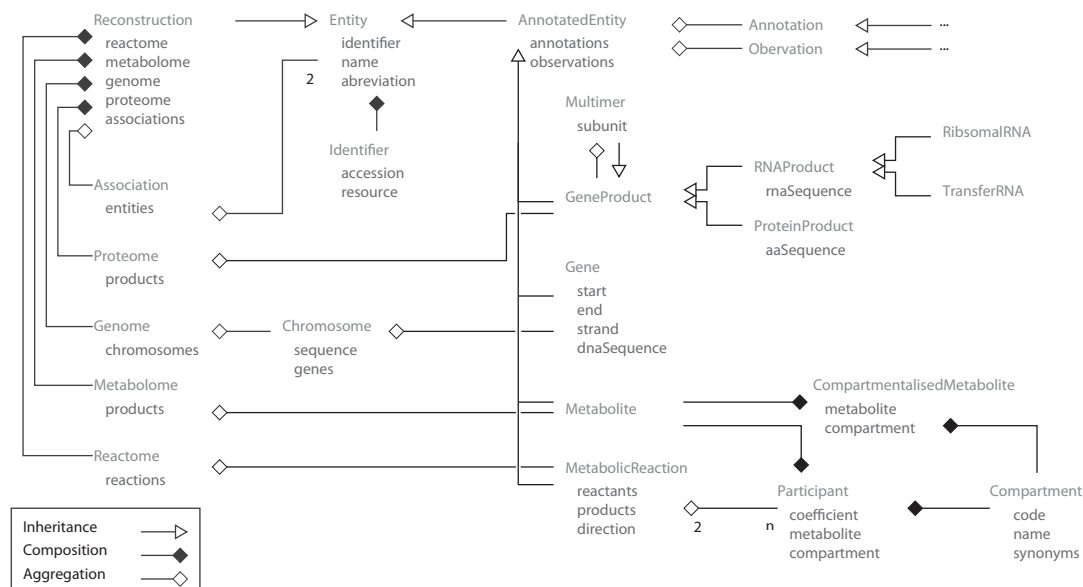


Figure 3.1: Object model representation and interactions. Names of objects are headed in blue with data members listed below. Unless specified aggregation is specified as a 1.. n relationship. Specific annotation and observations types are not listed.

3.2 Methods

3.2.1 Data representation

The library and desktop application are implemented in the Java programming language (version 1.6+). To describe the reconstruction, an object-orientated programming model is used^[174]. In this model an object is composed of data and operations on that data.

Entities

The components of a reconstruction are stored in several core objects (Fig. 3.1). The base object is an `Entity`, that contains a name, abbreviation, identifier, and confidence rating.

An `AnnotatedEntity` is derived from an `Entity` and extends the functionality to hold arbitrary data. Each `AnnotatedEntity` holds a set of annotations and observations. Operations are provided that can add, remove, or query these from the entity. The

```

...
<species id="m_atp_c" name="ATP" compartment="c"/>
<species id="m_atp_e" name="ATP" compartment="e"/>
<species id="m_atp_p" name="ATP" compartment="p"/>
...

```

Figure 3.2: Metabolites present in multiple compartments are defined as distinct entries in SBML.

concept of having annotations and observations was derived from the GenDB genome annotation system^[175] but here the definitions are less strict.

Annotations have either been manually entered, imported, or extrapolated. Whilst observations are numerous and usually the result of a computation. Observations can be considered as supplementary or supporting annotations. In practise the observation implementations have only been used to encode results from sequence homology searches.

Reconstruction The Reconstruction object is the top level object. It composes multiple *-omes* that each aggregate a specific type of AnnotatedEntity (Fig. 3.1). The Reconstruction provides high-level management and access to all other entities.

Metabolites The Metabolite object provides operations that simplify access to formal charge and chemical structure annotations. A single metabolite entity may appear in multiple compartments, as they are encoded as a property of the reaction. This is different to the view in the model. To enable simulation of transport, metabolites in different compartments must be distinct entities (Fig. 3.2). The representation used in Metingear is more convenient for annotation as only a single entry must be updated.

Reactions Reactions are encoded as a collection of reactants and products. Reactants are consumed, products are produced. The direction the reaction runs can be indicated as forward, backwards, bidirectional (reversible), or unknown. The reactants and products are collectively referred to as Participants. A participant is composed of a metabolite, stoichiometric coefficient (multiplier) and a compartment. Discrete values for available compartments (Tab. 3.1) were derived from existing reconstructions^[2].

Table 3.1: Enumeration of compartment definitions used in Metingear. The definitions and synonyms were derived from existing reconstructions and the Gene Ontology (GO).

Name	Code	Synonyms	GO Term
cytoplasm	c	in, cytosol, internal species, cytoplasm matrix, cell	GO:0005737
periplasm	p	periplasmic space	GO:0042597
golgi	g	golgi apparatus, golgi complex, golgi ribbon	GO:0005794
lysosome	l		GO:0005764
glycosome	y		GO:0020015
glyoxysome	o or w		GO:0009514
peroxisome	x	peroxisomal, peroxisome vesicle	GO:0005777
mitochondrion	m	mitochondria, mitochondrium	GO:0005739
nucleus	n	cell nucleus	GO:0005634
endoplasmic reticulum	r	er	GO:0005783
chloroplast	h		GO:0009507
apicoplast	a		GO:0020011
plastid	s		GO:0009536
thylakoid	t	photosynthetic membrane	GO:0009579
vacuole	v	vacuolar carboxypeptidase Y	GO:0005773
flagellum	f		GO:0019861
golgi membrane	gm		GO:0000139
mitochondrial membrane	mm		GO:0031966
nuclear membrane	nm		GO:0031965
endoplasmic reticulum membrane	rm	er Membrane	GO:0005789
vacuolar membrane	vm		GO:0005774
peroxisomal membrane	xm	peroxisome membrane	GO:0005778
plasma membrane	pm	plasmalemma, juxtamembrane, inner endospore membrane, cytoplasmic membrane, cell membrane, bacterial inner membrane, plasma membrane lipid bilayer	GO:0005886
unknown	unk	<i>an empty string is considered unknown</i>	
extracellular	e	extracellular region, Out, Extra_Organism, External_Species, External	GO:0005576
boundary	b	model boundary	

Genes and Gene Products Metingear is primarily focused on the metabolites and reactions of reconstructions. There is however some support for sequence data. The Gene and GeneProduct objects use BioJava^[176] to hold DNA, RNA and peptide sequences. Each Gene also holds a start and end position of the gene on a Chromosome. Each reconstruction contains a single Genome holding one or more Chromosomes. The Multimer object represents a gene product composed of multiple subunits.

Relationships Within the Reconstruction there are several relationships that can be defined. Some relationships are explicitly stored in the data model; for example, each reaction knows the metabolites involved. It is also desirable to know other relationships including what reactions a metabolite participates in and which gene products (enzymes) are required for a reaction.

Including this information in the data on each object may require cyclic references and leads to complications when storing and modifying the data. In particular an editing system must track and maintain relationships within and between each object. Implicit relationships between two objects are stored as an indirect link and managed by the top level reconstruction object (via Association). Separating these relationships from the data is more flexible and enables the addition of new relationships when required.

Identifiers The Identifier object stores an accession and resource. Identifiers are used as a primary data value on an Entity but are also used by cross-reference annotations. The majority of identifiers use the MIRIAM registry allowing access to a resolvable URL. Identifier types are organised in a hierarchy such that it is possible to determine whether an identifier is specific to a chemical, reaction, literature, etc. These abstract identifier types are used to provide richer annotations.

Annotations The annotations allow the flexible storage of values on reconstruction entities. Like identifiers, annotations are also arranged in a hierarchy. Operations on the annotated entity provide access to specific or generic annotation types (Fig. 3.3). Accessing a generic type (e.g. Name) returns all annotations of that type present on an entity. The hierarchy also allows the same user interface components to be reused for different annotation types. The context of an annotation defines the type of entity to which it can be added (Fig. 3.3).

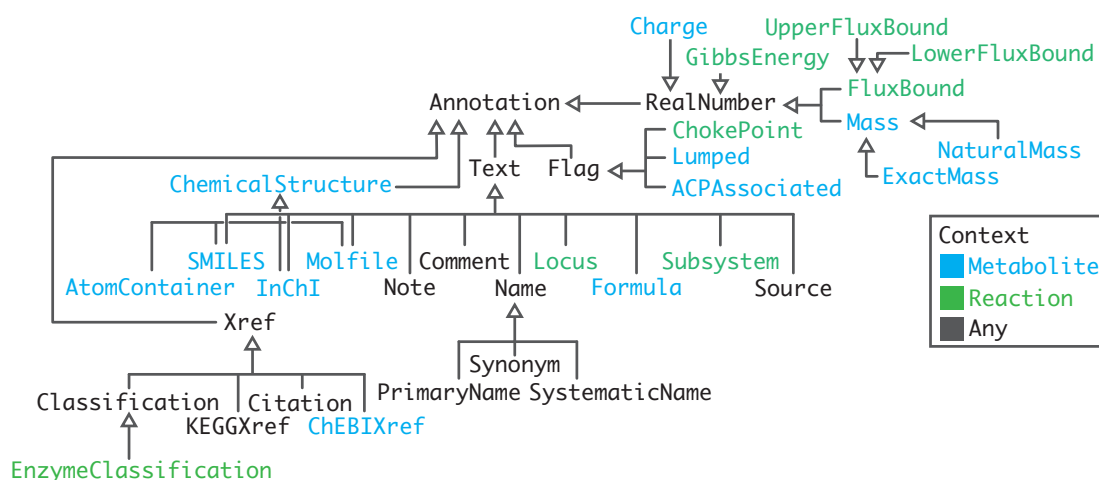


Figure 3.3: Hierarchy of annotation types; some names are changed for brevity. The arrows point from a more specific annotation to a less specific (generic) type.

3.2.2 Data import and export

Binary

Internally Metingear uses a binary format to store reconstructions. This format provides efficient loading and storage with minimal interruption to the user.

The efficiency means it is practical to store and load entire metabolic data resources such as MetaCyc, SEED and MNXref. Each resource can then be manipulated and modified no different to the smaller organism specific reconstructions. As demonstration of the efficiency, the SBML file for the most recent consensus reconstruction of human metabolism^[64] is approximately 32,505,856 bytes, while the Metingear binary is only 2,831,160 bytes. Loading from the binary formats takes ~1 second and loading from SBML takes ~20 seconds¹.

To allow backwards compatibility between software versions, the architecture defines a reader and writer (a marshal) for each entity, annotation, and observation type. When an object is modified, new marshals are created for that type. Each marshal defines the lowest version compatibility for which it can be used. The latest version of a reader (prior to the input version) is then used to read the binary data (Fig. 3.4).

¹Conversion from XML to JSBML takes 11 seconds, the conversion JSBML to MDK takes 9.3 seconds

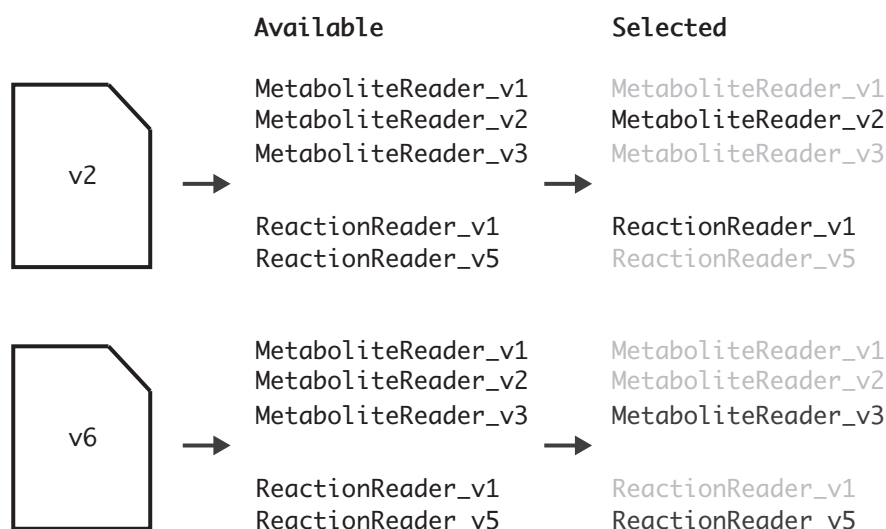


Figure 3.4: For a given input version, an appropriate reader is selected to interpret the binary and create object representations.

Systems Biology Markup Language (SBML)

Loading and storage of SBML is accomplished by converting the representation to the JSBML library. As a model format, SBML lists metabolites in different compartments as discrete species elements. When imported, metabolite species with the same name are merged into a single entry. Metabolites that have been assigned different names in compartments can be merged after import. The export of SBML does the reverse and creates separate species for metabolites in separate compartments. Translation of the SBML compartments is attempted automatically using the compartment definitions (Tab. 3.1). If resolution of compartment was not possible, the program prompts for user input during parsing. RDF annotations are used to encode MIRIAM resource identifiers and InChI representations. The notes element of SBML is also utilised and loaded as an annotation.

Tabular

Reconstructions can be imported from a tabular format provided as a Microsoft Excel workbook. Many existing reconstructions have been distributed in this format and it is useful to import as the tables may contain more information than the respective SBML.

Please confirm the appropriate columns in the reaction sheet. Please note that the data bounds should **not** include the table header

Start Row End Row

Identifier/Abbreviation Name/Description

Reaction Equation Classification (EC/TC Number)

Subsystem/Reaction type Source/Reference:

Locus

select data range (red arrow pointing to Start Row and End Row)

select column index (red arrow pointing to Reaction Equation)

select arrow to select more fields (red arrow pointing to Extra Columns)

description updates (green dashed line pointing to Reaction Equation)

grey row ignored (green dashed line pointing to Row 1)

	Abbreviation	Description	Equation	Subsystem	Classificati...	F	Source
1							
2	DKMD2K	1,2-dihydr...	[c] : 12d3k...	Amino acid...		BG13285	Ashida et a...
3	DKMD3M	1,2-dihydr...	[c] : 12d3k...	Amino acid...		BG13285	Ashida et a...
4	DM1PE	2,3-diketo...	[c] : dkmp...	Amino acid...		BG13282	Ashida et a...
5	HKM1PP	2-hydroxy...	[c] : 2h3k5...	Amino acid...		BG13283	Ashida et a...
6	MDRPD	5-Methylth...	[c] : 5mdru...	Amino acid...		BG13284	Ashida et a...
7	MTRI	5-methylth...	[c] : 5mdr1...	Amino acid...	EC-5.3.1.23	BG13278	Ashida et a...
8	IG3PS	Imidazole-...	[c] : gln-L...	Amino acid...		BG12600	Bauer et al.,...

Figure 3.5: Import from a tabular format requires configuration. The row ranges and column locations are selected and a preview is provided for convenience.

Although SBML defines a standard way of encoding information, the content may still be varied (Apdx. B.2). In particular, the formula and charge of a metabolite are often omitted from SBML. In SBML, there is no standard way to specify the molecular formula and the charge of a metabolite. Charge could originally be encoded as an attribute of the species but its use has since been deprecated^[177]. It is also possible that the cross-references are included in the table but not in the SBML. The popular model-SEED pipeline exports draft reconstructions as Excel workbooks and SBML. The workbooks include cross-references to KEGG whilst the SBML does not.

Lossless import of unstructured data from Excel workbooks is challenging. Existing applications require precise table structure or field definitions. Instead of defining a required format or naming, Metingear allows a user to select the column and row range containing the required information (Fig. 3.5). The locations and ranges can be set programmatically, allowing automated input.

At the absolute minimum the only required column is the reaction equation. The remaining columns in the reaction table and entire metabolite table are optional. Reaction columns that can be defined include: identifier, name, classification, subsys-

tem, source, locus, direction, Gibbs energy (and error), and flux bounds. Columns of metabolite table are: identifier (abbreviation), name, charge, formula, compartment and cross-references (KEGG, ChEBI or PubChem-Compound). Once the reaction equation is interpreted metabolites are linked to reactions by finding a matching identifier or name. If no match was found a warning is reported. Multiple reactions can be handled by importing the document more than once and specifying multiple tables.

Parsing reaction equations

Reactions are specified in the tabular format with a textual equation. At its simplest, the equation states that a collection of metabolites is transformed, producing another collection. The direction of the reaction and reversibility is indicated with an arrow between the two collections, stoichiometry with a coefficient, and compartmentalisation with a predefined code.

The method used to parse reactions was iteratively refined from trial and error on published reconstructions. The process currently works by splitting up the input equation and processing each side and their participants separately. The grammar is formalised in Listing 3.1.

Only single step reactions are interpreted but the procedure naturally extends to multi-step reactions. For brevity coefficients of a unit value and default compartments are normally omitted. The default compartment for a reaction is set when specified as a prefix, if missing it is presumed to be the cytosol. Figure 3.6 shows an example of the expansion of these implicit terms for a reaction equation input.

Once parsed, the description of metabolite is looked up in the metabolite table. If the description of the metabolite is not found a warning is issued. During evaluation, a warning was issued when importing the *iBsu1103*^[143] reconstruction from a tabular format. The warning identified that three reactions (rxn003194, rxn03436 and rxn08764) referenced an unresolved metabolite (cpd00498). Further inspection revealed the metabolite description was missing from the metabolite table.

The described parsing is not comprehensive and there are several ambiguities that were encountered during testing. Firstly, participants may not be separated from the addition symbol by a space, examples include 'YGH+FHD' and '35FGD+1400DH'. This input is ambiguous as to whether single or multiple metabolites are indicated.

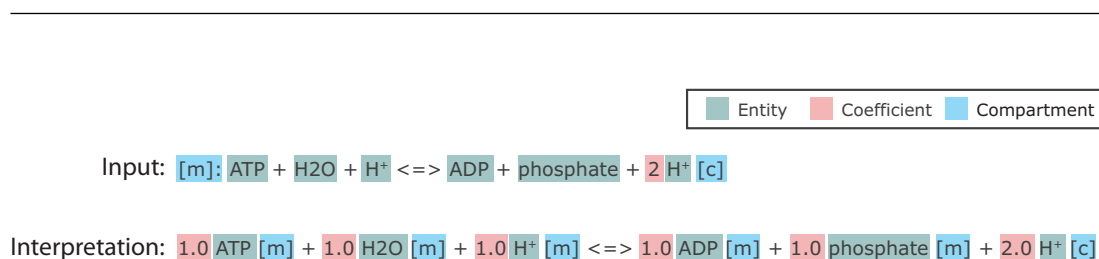


Figure 3.6: Expansion of implicit terms when parsing reaction equations.

Currently these are both interpreted as a single entity – this is required to allow for entry of ions that can include an addition symbol, for example ‘H⁺’.

Additional information can be associated with metabolites; for protons an example is seen in the reconstruction of *Halobacterium salinarum*^[178] where the reaction R90124 indicates one proton as being associated with the proton motive force (pmf)¹.

Finally, reactions may not include real numbers for coefficients but instead include a polynomial with one or more variables ($2n$, $3m$, etc). Handling of these reactions is not supported but has only been encountered in database flat-files (KEGG) and not reconstructions. The exceptions listed here are relatively rare and as reactions can be modified, any mistakes can be corrected manually.

Listing 3.1: Reaction equation grammar in a modified Backus-Naur Form

```

<direction>      ::= "<" ("-" | "=")* ">"
                  | "<" ("-" | "=")*
                  | "="+
                  | ("-" | "=")* ">"
<compartmentnet_code> ::= "c" | "e" | "n" | ...
<compartment>    ::= "[" <compartmentnet_code> "]"
<coefficient>    ::= <number> | "(" <number> ")"
<and>            ::= <space> "+" <space>
<entity>         ::= (!<and>) +
<participant>    ::= [<coefficient>] <entity> [<compartment>]
<side>           ::= <participant> [<and> <participant>]+
<reaction>       ::= [<compartment> ":" ] [<side>] <direction> [<side>]

```

KGML, Cytoscape and BioPax

KEGG exports pathways in a custom KGML format that can be imported directly. The stoichiometric matrix for a reconstruction can be exported as a text table or a binary coordinate list that compresses sparse matrices. A .sif file can be created for

¹L-Citrulline [e] + H⁺ (pmf associated) => L-Citrulline + H⁺

visualising the network in Cytoscape^[179]. The .sif export can duplicate highly connected currency metabolites, above a set threshold, for improved visualisation. The BioPax^[25] format is not directly supported but tools exist for conversion to SBML¹.

Genome and sequence data

Genomes from the European Nucleotide Archive (ENA)^[180] and peptide sequences from a FASTA file can be imported. The ENA XML format provides the genome, coding sequences and functional annotations. The import creates genes and gene products in the reconstruction and adds cross-references for any specified annotations.

3.2.3 Data resources

Access to and searching metabolic data resources is accomplished through an abstract set of service descriptions. The services provide a common layer from which information can be retrieved and queried. Each service is self describing and if required, linked with a MIRIAM resource description. This allows the application to be extended and operate independently of a specific resource. The framework is similar to that used by libAnnotationSBML and SAINT. The key differences are: services are dynamically discovered and data can be indexed locally for faster retrieval.

The dynamic loading allows multiple access points to exist for the same resource; the best access point is chosen and utilised. This configuration allows fast access when needed but does not disable functionality if a resource has not been indexed. The setup proved beneficial when both the KEGG and PubChem resources changed their web services protocols. A new service could be written and used in conjunction with the existing protocols until the old service was retired.

Service types

Services are described by a collection of abstract interfaces. Each service type specifies access or search to a specific type of information such as a name. The search and access descriptions are separate to describe services that only provide one of the operations. A service instance is a realisation of one or more descriptions for a specific data

¹See <https://github.com/johnmay/metingear/wiki/Import#ibp>

Table 3.2: Overview of service implementations. The access and search to each data field is split as to define service that only provide one or the other. Local services are more flexible as the queries are controlled.

		ChEBI		KEGG		MetaCyc		HMDB		LIPID MAPs		PubChem		UniProt	
		local	remote	local	remote	local	remote	local	remote	local	remote	local	remote	local	remote
Preferred Name	Access	•	•	•	•	•		•		•		•			
	Search	•	•	•	•	•		•		•					
Synonym	Access	•	•	•	•	•		•		•		•			
	Search	•	•	•	•	•		•		•					
IUPAC Name	Access	•	•		•	•		•		•		•			
	Search	•	•		•	•		•		•		•			
Name	Access	•	•	•	•	•		•		•		•			
	Search	•	•	•	•	•		•		•		•			
Formal Charge	Access	•						•							
	Search	•						•							
Molecular Formula	Access	•		•	•			•				•			
	Search	•		•	•			•							
Structure	Access	•	•	•	•	•		•		•		•			
	Search Identity	•		•		•		•		•					
	Search Substructure	•		•		•		•		•					
Cross-references	Access	•												•	
	Search	•												•	
Reaction	Access			•											
	Search			•											

resource. As noted previously, there may be multiple instances for the same resource. Table 3.2 summarises the local and remote instances provided in Metingear.

Programmatic access

Internally the library interacts with the services through a manager. On initialisation the manager loads service descriptions using the Java Service Provider Interface (SPI) framework^[181]. The manager determines availability and provides a requested service instance for an identifier type (Listing 3.2). The manager can also create a custom service instance that proxies calls to other services. In Listing 3.3, the invoker defines

a new service type that provides both name and structure access. A service is then requested for ChEBI entries and the manager looks up an instance of each required service and creates the Java bytecode that wires these together into a usable service instance.

Listing 3.2: Requesting a service for an identifier

```
// request a service to access preferred names of ChEBI identifiers
ServiceManager mgr = DefaultServiceManager.getInstance();
PreferredNameAccess srv = mgr.getService(ChEBIIdentifier.class,
                                         PreferredNameAccess.class);

// access the preferred name of "CHEBI:15422"
String name = srv.getPreferredName(new ChEBIIdentifier("CHEBI:15422"));
```

Listing 3.3: Defining and creating a custom service

```
// define a custom service description
interface MyService extends NameService, StructureService {
}

// request the creation of a service instance that provides this functionality
// for ChEBI identifiers
ServiceManager mgr = DefaultServiceManager.getInstance();
MyService srv = mgr.createService(ChEBIIdentifier.class,
                                  MyService.class);

Identifier id = new ChEBIIdentifier("CHEBI:15422");
String nme = srv.getPreferredName(id);
IAtomContainer cpd = srv.getStructure(id);
```

Local services

A remote resource may not expose all the stored data via a remote service. For example, the KEGG and MetaCyc databases do not provide structure search. Latency and query limits (e.g. 20 per second) mean the access to remote resources can be slow. Indexing the data locally allows for both customisation of data and queries as well as providing faster retrieval.

Depending on the type of data, a key-value store or a relational database is used. Apache Lucene^[182] is currently used to store the key-value pairs as the index was useful for name searching. HyperSQL^[183] is used to store reactions that require a relational representation.

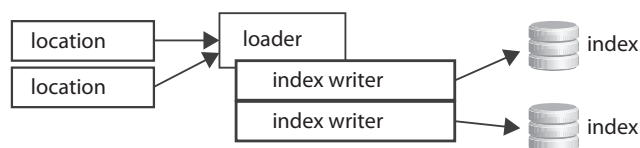


Figure 3.7: The service loading architecture for creating a local data index.

Local indices are created with a *loader*. A loader parses an input, specified by a *location*, and uses one or more standard *index writers* to create an *index* (Fig. 3.7). The purpose of the locations is to provide the contents to the loader to parse. Types of locations include: local file, local folder, remote file, compressed local file, compressed remote file, etc. The locations also include a description of the expected input. The index writers provide a standard means of creating the local index that can then be reused by other loaders. Standardising the index writing also means the service instances (which load the index) are simplified through code reuse.

The loaders to create the resources can be invoked through a graphical interface:

ChEBI				
				ChEBI Chemical Structures updated 06-Feb-14 14:42
				ChEBI Names updated 03-Feb-14 19:56
				ChEBI Data updated 03-May-13 13:26
				ChEBI Cross-references updated 18-Mar-13 17:38
KEGG				
				KEGG Compound updated 16-Sep-13 10:28
				KEGG Compound (Structures) updated 29-Apr-13 10:21
				KEGG Reaction updated 29-Apr-13 11:48

Each row in the interface represents a local index and several actions are available on the index. From left to right these actions are; delete the index, revert to the previous index, configure, and update. The configure option opens up a form where the input locations can be specified. If a location is available via public FTP, the form is already completed. The update button is only enabled if all the required locations have been specified. When the required file is not publicly available, as is seen with KEGG, the loader must be configured before the index can be updated.

3.2.4 Graphical interface

The Metingear desktop application provides a graphical user interface and utilities that assist in the handling and modification of metabolic reconstructions. This section summarises the key features of interacting with the interface. A detailed manual with tutorials is available at <https://github.com/johnmay/metingear/wiki>.

The user-interface is divided into three main sections: entity table, entity inspector, and side bar (Fig. 3.8). The entity table (yellow) lists all genes, gene products, reactions, or metabolites in the active reconstruction. Only one type of entity is listed at a time. The type of entity can be switched by selecting an entity, toggling the view control, or by selecting the entity type in the sidebar (blue). The detail of a selected entity is displayed in the entity inspector (green). Multiple reconstructions can be open during a session but there can only be one active reconstruction. Entities can be moved between open reconstructions by copying or dragging a selection from the table. Changes made to the attributes and relations of entities are tracked and can be undone and redone during a session.

Table view Each entity type has a separate entity table that displays a synopsis for all entities of a given type in the reconstruction. At a minimum, the table displays the accession (primary identifier), abbreviation, name and a rating. The other columns are specific for each entity type (Fig. 3.9). The abbreviation, name, and rating can be modified directly in the table by double clicking the cell. A selected entry is displayed in the inspector view.

Entity inspector The inspector displays specific details on a selected entry and exists in two modes, *view* and *edit*. The accession, name and abbreviation are indicated at the top of the inspector and can be modified directly when in edit mode. Annotations attached to an entity are listed on the right in an interactive table and these can be modified or deleted in edit mode. Relations to other entries are displayed in the centre; a metabolite lists the reactions it participates in (Fig. 3.10a). Observations on an entity are listed below the synopsis and annotation sections in the inspector (not shown).

A synopsis for an entity is also displayed and for a metabolite this consists of a selected structure depiction and the molecular formula (Fig. 3.10a). The reaction inspector lays out the name and a structure depiction of each metabolite (Fig. 3.10b). The name of

Accession	Abbreviat...	Name	Generic	Cross-reference	Molecular ...	Validity	Match	Rating	Virtual	ACP Associated
M/1544...	alac-S	(S)-2-Acetolactate	No	CHEBI:18409, C...	C5H7O4	✓	★	★		
M/1544...	3mop	(S)-3-Methyl-2-oxopenta...	No	CHEBI:35146	C6H9O3	✓	★	★		
M/1544...	dhor-S	(S)-Dihydroorotate	No	CHEBI:30864	C5H5N2...	✓	★	★		
M/1543...	mmcoa-S	(S)-Methylmalonyl-CoA	No	CHEBI:15466, C...	C25H35...	✓	★	★		
M/1543...	12ppd-S	(S)-Propane-1,2-diol	No	CHEBI:29002	C3H8O2	✓	★	★		
M/1543...	12dgr_EC	1,2-Diacylglycerol (E.coli)...	No		C1836H...	?	★	★	Virtual	
M/1544...	dhna	1,4-Dihydroxy-2-naphtho...	No	CHEBI:11173	C11H7O4	✓	★	★		
M/1544...	15dap	1,5-Diaminopentane	No	CHEBI:18127	C5H16N2	!	★	★		
M/1544...	2cpr5p	1-(2-Carboxyphenylamin...	No	CHEBI:58613, C...	C12H13...	!	★	★		
M/1544...	prfp	1-(5-Phosphoribosyl)-5-[...	No	CHEBI:58435, C...	C15H21...	!	★	★		
M/1544...	prbamp	1-(5-Phosphoribosyl)-AMP	No	CHEBI:18374	C15H19...	!	★	★		
M/1544...	prbatp	1-(5-Phosphoribosyl)-ATP	No	CHEBI:18263	C15H19...	!	★	★		

(a) Metabolites (sorted by name)

Accession	Abbreviat...	Name	Reversibili...	Equation	Enzyme Cl...	Subsystem	Compart...	Rating
R/2426...	ADD	adenine ...	→	Adenine [c] + H+ [c] + H2O...	3.5.4.2	Nucleoti...	{c}	★
R/2426...	L-LACD2	L-Lactat...	→	L-Lactate [c] + Ubiquinone-...	1.1.2.3	Oxidativ...	{c}	★
R/2426...	L-LACD3	L-Lactat...	→	L-Lactate [c] + Menaquinon...	1.1.2.3	Oxidativ...	{c}	★
R/2426...	ATPS4r	ATP synt...	⇌	ADP [c] + 4.0 H+ [e] + Pho...	3.6.3.14	Oxidativ...	{c, e}	★
R/2426...	CRNBTCT	gamma-...	⇌	gamma-butyrobetainyl-CoA...		Oxidativ...	{c}	★
R/2426...	CRNCBCT	crotonob...	⇌	L-Carnitine [c] + crotonobet...		Oxidativ...	{c}	★
R/2426...	CRNCDH	Carnityl-...	⇌	Carnitiny-CoA [c] ⇌ crotono...		Oxidativ...	{c}	★
R/2426...	CYTBD	cytochro...	→	2.0 H+ [c] + 0.5 O2 [c] + U...		Oxidativ...	{c, e}	★
R/2426...	CYTBO3	cytochro...	→	2.5 H+ [c] + 0.5 O2 [c] + U...		Oxidativ...	{c, e}	★
R/2426...	LDH_D2	D-lactat...	→	D-Lactate [c] + Ubiquinone-...	1.1.2.4	Oxidativ...	{c}	★
R/2426...	DMSOR1	Dimethyl...	→	Dimethyl sulfoxide [c] + Me...		Oxidativ...	{c}	★
R/2426...	DMSOR2	Dimethyl...	→	2-Demethylmenaquinol 8 [c...		Oxidativ...	{c}	★

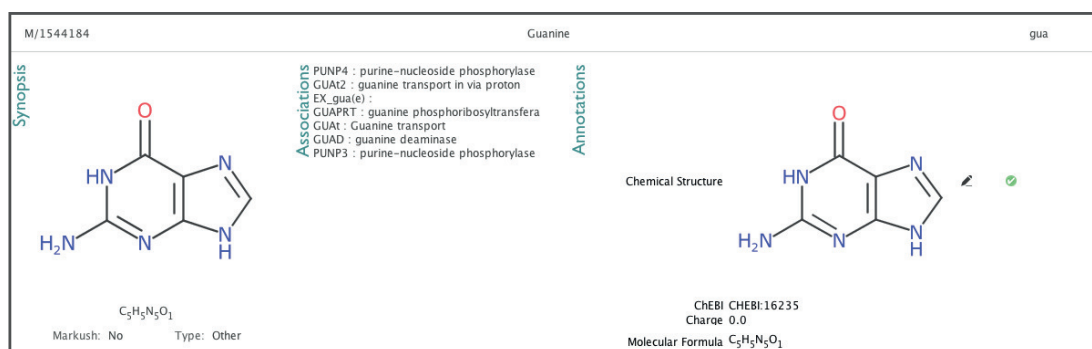
(b) Reactions

Figure 3.9: Table display of metabolites and reactions from the *iJR904* reconstruction^[167].

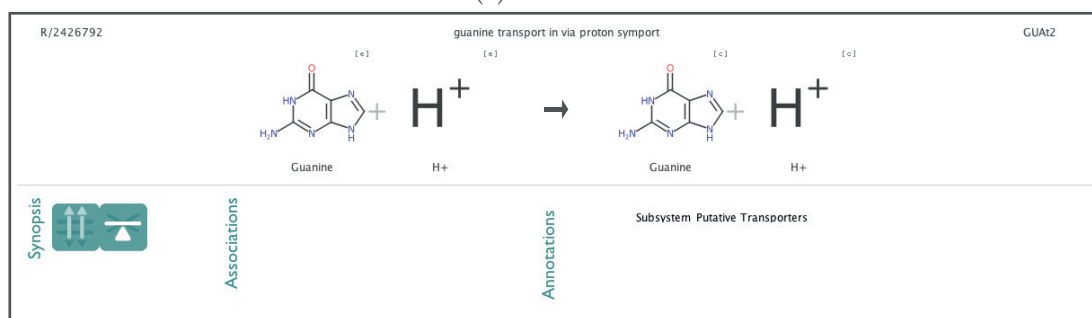
each metabolite in the reaction selects and navigates to the metabolite in the interface. The reaction inspector also indicates whether a reaction is a transporter and the general type. The mass balance is indicated by a set of scales icon that tips to the side with more mass, while charge balance is not currently indicated.

Creation of entities Entries are automatically created when a reconstruction is imported but additional entries can also be manually created and modified. To assist the creation of new entries input fields provide auto-completion. The auto-completion of entries is achieved with either a Lucene index or a prefix search using a ternary search tree^[184].

Creating a reconstruction requires that organism information is specified. By typing parts of an organism name or code the dialog offers suggestions from the Uniprot curated species list^[185] (Fig. 3.11). When entering a reaction equation, the input suggests names of metabolites listed in the active reconstruction and ChEBI. ChEBI is also used to provide an auto-complete when creating metabolites; suggested metabolite names are cycled with the *tab* key.



(a) Metabolite



(b) Reaction

Figure 3.10: Inspectors displaying additional information on guanine and a guanine symporter.

Create a new reconstruction

Internal Reconstruction Identifier:

Organism Code (e.g. ECOLI):

Organism Name:

Taxon Code:

Kingdom (e.g. B,E,V,A):

- Brevibacterium epidermidis
- Epidermophyton floccosum
- Epidermophyton magnoliae
- Epidermophyton stamfordianum
- Porcine epidemic diarrhea virus (strain CV777)
- Porcine epidemic diarrhea virus (strain Br1/87)
- Staphylococcus epidermidis
- Staphylococcus epidermidis (strain ATCC 35984 / RP62A)
- Staphylococcus epidermidis (strain ATCC 12228)

Figure 3.11: Creation of a new reconstruction requires information on an organism. When a name or code is entered a drop-down menu lists potential inputs.

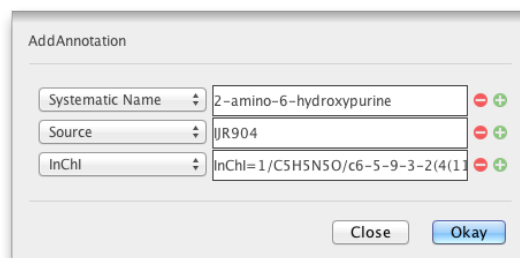


Figure 3.12: Manual annotations can be added to a selected entity. Given an input field, a user enters the content and selects the annotation type. On adding the annotation, the input is transformed into the specified annotation type.

Manually adding annotations Annotations can be manually created and added to entities. The annotation input requires the selection of the annotation type and entry of the value (Fig. 3.12). Using the annotation context, only annotations relevant to the active entry can be select.

Searching Whilst editing a reconstruction a Lucene index is continuously updated providing instant search of entries in the interface. The search field (Fig. 3.8) supports loose input, such as “Sucrose” and more specific term based queries, such as “Name:Sucrose + Type:Metabolite”.

Collections A selected set of entities can be isolated into a named collection so that they can be displayed and modified in a separate table. The collection currently only remains visible for the session.

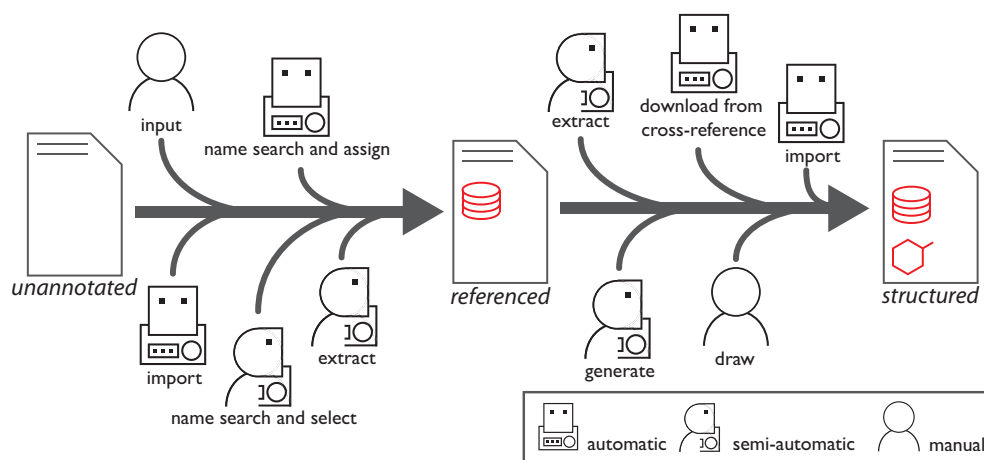


Figure 3.13: Overview of available annotation methods in Metingear. A metabolite can be annotated with a cross-reference or chemical structure. A referenced metabolite can directly lead to one with a chemical structure by propagating annotation from the reference.

3.3 Results and discussion

Metingear has been used to annotate genome-scale metabolic reconstructions for alignment (Chap. 2). This section describes how precise annotations can be assigned and discusses examples from published reconstructions. The annotation of a metabolite is to identify it with either an indirect cross-reference or chemical structure annotation is added. These annotations can be added in a manual, semi-automated, or fully-automated manner (Fig. 3.13).

3.3.1 Cross-references

A cross-reference describes an entry in a resource using an accession and a resource identifier. With a valid cross-reference, annotations from the resource can be retrieved using the service framework and propagated on to the entity. It is therefore useful to be able to add, import, and find cross-references for each entity.

Input Cross-references can be manually specified by creating a new annotation. The annotation requires input of the accession and selection of the data resource; metabolite cross-references can be modified directly in the table (Fig. 3.14). When the accession is entered the text is scanned for matching MIRIAM registry patterns and only matched resources are available for selection.

M/1544181	xmp	Xanthosine 5-phosphate	No	CHEBI:5...	C10H11...				★
M/1544182	hxan	Hypoxanthine							★
M/1544187	dudp	dUDP							★
M/1544188	xtsn	Xanthosine							★
M/1544189	bbtcoa	gamma-buty...							★
M/1544183	imp	IMP							★
M/1544184	gua	Guanine	No	CHEBI:1...	CSH5N5O	✓			★

Figure 3.14: Adding a cross-reference for guanine. Metabolite cross-references can be modified in the table directly. On input, the accession is checked against the MIRIAM registry and the resource type is inferred. The resource type can be manually adjusted.

Table 3.3: Cross-references flat-file specifying the target entity (abbreviation), accession and resource type.

Abbreviation	Accession	Resource
atp	META:ATP	BioCyc
atp	CHEBI:15422	ChEBI
atp	C00002	KEGG COMPOUND
atp	5957	PubChem-Compound
gdp	6830	PubChem-Compound
gtp	6022	PubChem-Compound

Import A cross-reference may be present from the import of reactions and metabolites. SBML species annotated with resolvable MIRIAM registry descriptions are imported as cross-reference annotations automatically. When importing from Excel, a user can specify a column that contains a KEGG COMPOUND, ChEBI or PubChem-Compound identifier.

Cross-references can also be imported from an external value separated flat-file and attached to existing entries. This import option allows bulk assignment of cross-references from a database export or external mapping tool. The input format requires a primary column (id, abbreviation, or name) that is used to pair the imported cross-references with existing entities in the active reconstruction.

The cross-reference can be specified as either a single column containing accessions from a single resource or as two columns specifying the accession and the resource type (Tab. 3.3). It is possible to mix accessions to different resources in the single column input but the non-specific semantics (e.g. just a number) of some identifiers make it impossible to infer the data resource.

Extract Reconstructions may include cross-reference information as text notes (Listing 3.4). Using pattern matching, information encoded in the notes can be extracted

Listing 3.4: Notes in SBML for *Salmonella typhimurium* LT2^[186].

```
<species id="M_malttr_p" name="Maltotriose" compartment="p" charge="0" >
  <notes>
    <html xmlns="http://www.w3.org/1999/xhtml" >
      <p>FORMULA: C18H32O16</p>
      <p>KEGG ID: C01835</p>
      <p>PubChem ID: 4954</p>
      <p>ChEBI ID: 27931</p>
    </html>
  </notes>
</species>
```

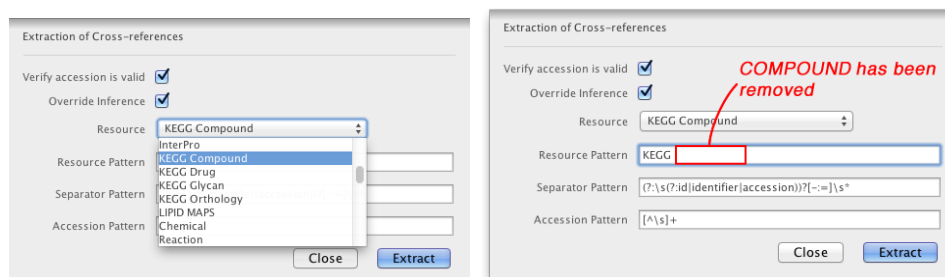


Figure 3.15: Configuring the extraction of annotations from free text notes.

and tagged as a specific cross-reference type.

As with the import from an external mapping file, identifier types can be specified or automatically inferred. This is important as a resource type may not precisely match the MIRIAM encoding. An example of this is seen in Listing 3.4 where identifiers are labelled as KEGG and PubChem. KEGG is used in reference to the KEGG COMPOUND and PubChem is used in reference to PubChem-Substance. Other reconstructions^[64] use PubChem to refer to PubChem-Compound. These ambiguities are problematic and difficult to resolve automatically. These examples emphasise the importance of the MIRIAM registry in unambiguously specifying a database identifier.

It is worth noting that the PubChem ambiguity was likely propagated from the KEGG COMPOUND entry that lists PubChem in reference to the deposited PubChem-Substance entry. This is also done in ChEBI and is a consequence of how data is submitted for integration in PubChem (via Substance).

To extract these identifiers from the notes the pattern matching must be modified (Fig. 3.15).

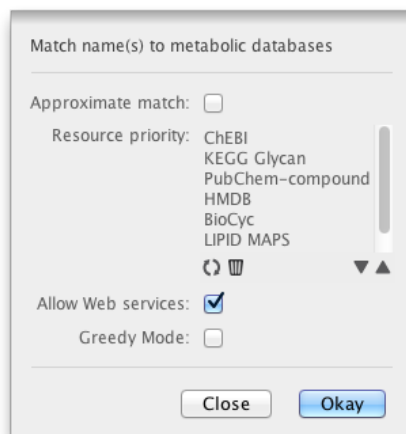


Figure 3.16: For the automated cross-reference using metabolite name a resource priority can be specified.

Name search The metabolite name can be used to search and identify entries in data resources which can then be added as cross-reference. As the same name may have been annotated to different entities in different resources, metabolite names are not seen as a robust form of identification^[187]. However, a name is often the only form of identification attached to a metabolite and offers the only starting point for annotation.

Assigning a cross-reference automatically from name first proceeds by selecting the data resources and their priority (Fig. 3.16). The priority determines the order in which each resource is searched. Resources can also be removed. If the process is set as greedy, references are retrieved from all resources regardless of priority. The non-greedy approach stops as soon as an acceptable reference is found. Each resource search provides a set of candidate identifiers for each resource that are then tested for acceptability. Names are acceptable if, after normalisation (Apdx. B.3), there are no differences. The approximate match returns more candidates to check but the equality check is not relaxed.

The resources are loaded dynamically using the resource framework described previously and any new service that provides a name search can be automatically used with the name search.

Names that differ by only one or two characters may be a valid match for a metabolite. These cross-references would be rejected by the automated procedure but can be accepted with user confirmation. In selection mode, each candidate is ranked by Levenshtein distance^[188]. The list of candidates is displayed for the user to choose (Fig.

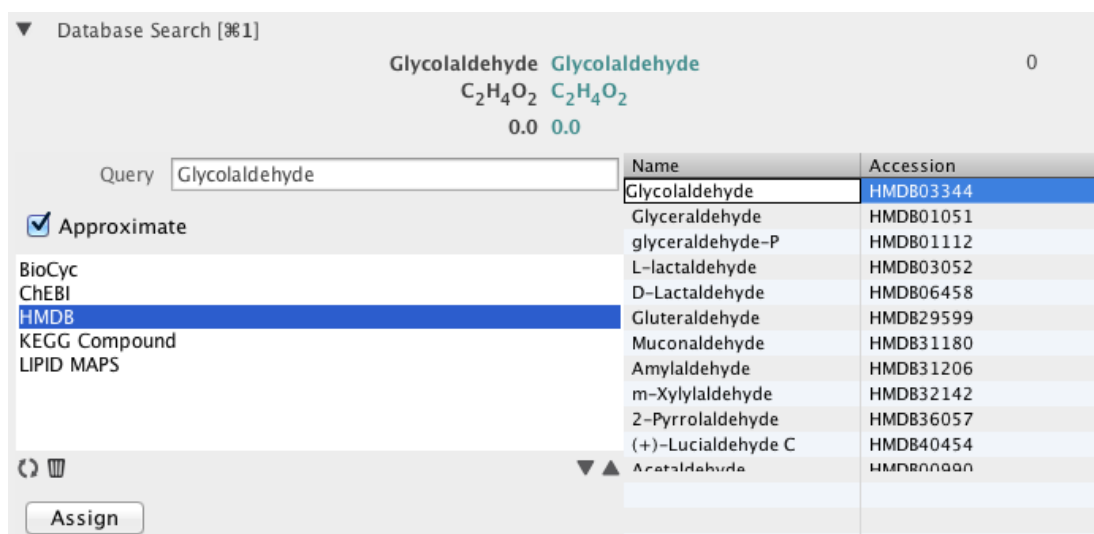


Figure 3.17: Selection of a cross-reference from HMDB^[45] for a metabolite named ‘Glycolaldehyde’. The input name can be modified and allows manual correction of typographical errors. The searched resource is selected from the list of local services.

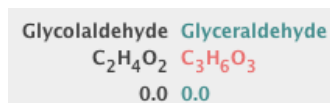


Figure 3.18: The quality of a cross-reference is indicated by inspecting name, formula, and charge differences. In this case we see the name has one difference but the formula has different heavy atom counts.

3.17).

Selecting a candidate indicates the quality of the cross-reference at the top of the dialog (Fig. 3.18). If available, the match of metabolite name, formula and charge is shown and the level of discrepancy is highlighted. The highlight colour is: green (okay), yellow (acceptable) or red (unacceptable). The name is indicated as okay when there are fewer than three characters different. The formula is acceptable if the number of protons differs by the specified charge.

3.3.2 Chemical structure annotations

Chemical structure annotations are stored in Metingear either as a CDK data type (AtomContainer) or a specific format (SMILES, InChI, or Molfile) annotation. These annotations can be propagated from a cross-reference or annotated directly. Direct

annotation allows structure representations to be input, imported, drawn manually, or generated when no cross-reference is available.

Listing 3.5: SBML notes for the species `M_11docrts1_c` in *Recon 2*^[64].

```
<notes>
  <body xmlns="http://www.w3.org/1999/xhtml" >
    <p>FORMULA: C21H30O4</p>
    <p>CHARGE: 0</p>
    <p>INCHI: InChI=1S/C21H30O4/c1-19-8-5-14(23)11-...</p>
    <p>EHMN_ABBREVIATION: C05488</p>
  </body>
</notes>
```

Transfer from cross-reference If a cross-reference annotation is available or has been added, the chemical structure can be automatically retrieved from the data resource and attached to the entity. The service framework is used to locate the structural representations for a specific identifier type. As with the name search, structures are retrieved following a set resource priority. Structures are iteratively retrieved for a given resource and if multiple cross-references are available for a resource, all structures will be attached.

Input, import or extract SMILES, InChI, molfile and CML formats can be input manually for each entry by copying the text representation. Similar to database identifiers, the SMILES and InChI linear notations can be imported from a mapping file or extracted from notes (Listing 3.5) using the same methods discussed for identifiers. Charge and formula annotations can be attached as well.

Generate Some metabolites may be difficult to find a cross-reference for but have an interpretable name. Structures for these metabolites can be generated automatically or with minimal user input.

Oligopeptides are often found in metabolic reconstructions that model peptidoglycan synthesis. A structure representation can be generated by assembling amino-acid subunits. A pattern match determines if the metabolite name contains three letter amino-acid codes and configures the generator. If unspecified, the L- stereo-form is chosen and the configuration can be changed if the interpretation was incorrect (Fig. 3.19).

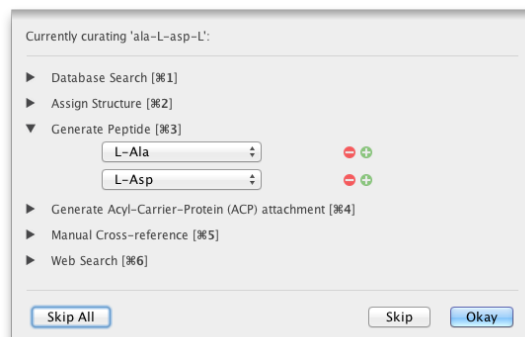


Figure 3.19: Oligopeptides can be generated by choosing the subunits.

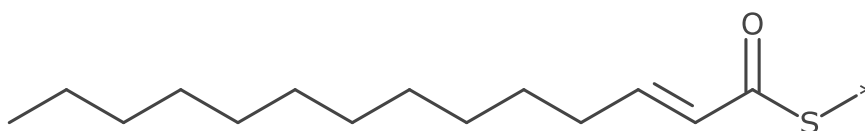


Figure 3.20: Generated structure for the metabolite Tetradenenoyl-ACP (n-C14:1ACP) from the *iJR904* reconstruction^[167].

Fatty acids and polyketides are attached to the acyl carrier protein (ACP) during synthesis. ACP attached metabolites can be represented as a (labelled) pseudo-atom in a chemical structure. Although data resources such as ChEBI, MetaCyc, and KEGG include structures for ACP attached fatty acids, small differences in how the ACP attachment is specified in the name means it may not be possible to locate a cross-reference in these resources.

If the name of the ACP attached metabolite is systematic, a structure can be generated using a chemical name to structure library (OPSIN)^[95]. Names may require an adjustment to be interpreted correctly (Tab. 3.4). OPSIN reports why a name cannot be interpreted, guiding required modifications.

The representation of ACP attached metabolites includes a pseudo atom adjacent to the sulphur (Fig. 3.20). This follows the convention used by ChEBI and KEGG. The representation in MetaCyc does not include the sulphur and the pseudo atom is adjacent to the carbonyl group. When the double bond placements are not specified in the name, OPSIN places it at the first available atom. This may be incorrect, and the metabolite may actually be representing a mixture (Fig. 3.20).

Inspection of metabolite entries in several reconstructions revealed the formula of the ACP attached metabolites did not always match the generated structures. The for-

Table 3.4: Examples of required naming adjustments to ACP attached fatty acids from the SEED database.

Id	Original	Corrected
cpd16911	cis-19-docosanoyl-ACP	cis-19-docosenoyl
cpd11542	15-methyl-3-hydroxy-hexa-decanoyl-ACP	15-methyl-3-hydroxy-hexadecanoyl
cpd16912	cis-19-cis-37-C38oyl-ACP	(19Z)-octatriaconta-19,37-dienoyl

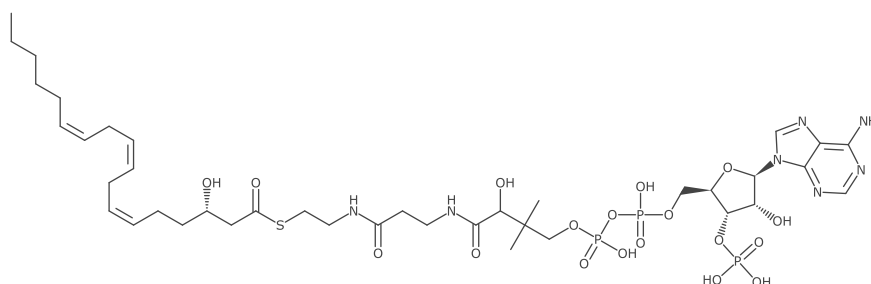


Figure 3.21: Generated structure for 3(*S*)-hydroxy-6*Z*,9*Z*,12*Z*-octadecatrienoyl-CoA (CE2438) from *Recon 2*^[64].

mula for these metabolites was found to include elements of the phosphopantetheine prosthetic group.

Draw The chemical structure of a metabolite can be drawn using an updated¹ version of the JChemPaint editor^[189]. The extended version includes newer features from CDK library for handling stereochemistry as well as name to structure using OPSIN. The editor can be use to manually create or modify structure annotations for metabolites.

As with the fatty acids attached to the ACP, OPSIN can be used to generate structures for other metabolites. The metabolite 3(*S*)-hydroxy-6*Z*,9*Z*,12*Z*-octadecatrienoyl-CoA (CE2438) is not annotated in the 2013 consensus human metabolic reconstruction. Modifying the prefix to (3*S*,6*Z*,9*Z*,12*Z*)-3-hydroxy-6,9,12-octadecatrienoyl allows OPSIN to generate the structure that can be attached to coenzyme A using the built-in JChemPaint editor (Fig. 3.21).

¹The primary version is not currently not maintained.

3.3.3 Verification

Verifying correctness of structural annotations is difficult without meticulous manual inspection. However, if the molecular formula and charge are present on a metabolite, they can be used as an indicator as to whether the structure annotation makes sense. Recall that the formula and charge may be imported from the tabular format or extracted from the notes.

The structural validity of an annotation is indicated in the interface table and inspector by a traffic light icon. This is the same system used in the name search when selecting a candidate from a list. A formula is either: good (green), acceptable (amber), bad (red) or unknown (grey) (Fig. 3.9a). An acceptable match is displayed when the number of hydrogens is correct considering a charge difference, this indicates the annotated structure may be at the wrong protonation state. The formula is not unique to a structure but serves as filter to quickly identify wrong annotations.

The verification could be used to automatically removed candidates during the name search. The advantage of the modular approach used in Metingear is that structures that don't match exactly may only need a small modification that can be made with the built-in structure editor.

3.3.4 Standardisation

The structural formula for a metabolite may exist in several distinct and equally valid representations. The multiple representations may be a limitation of the graph model or constitutional isomers that can interconvert, tautomers. Alternatively a structural formula may be at a different protonation state or have unspecified stereochemistry.

Metingear currently includes procedures to automate the standardised representation of tautomers. Procedures for stereochemistry and protonation are to be integrated in future releases.

Two types of tautomerism are handled by Metingear: hydrogen migration and ring-chain. For hydrogen migration the Sayle and Delany algorithm^[190] (implementation discussed in Chapter 5) is used to enumerate or standardise representations. The standardisation generates a canonical tautomer; the process is non-interactive and can be applied to all metabolites in a reconstruction. The enumeration of tautomers, lists

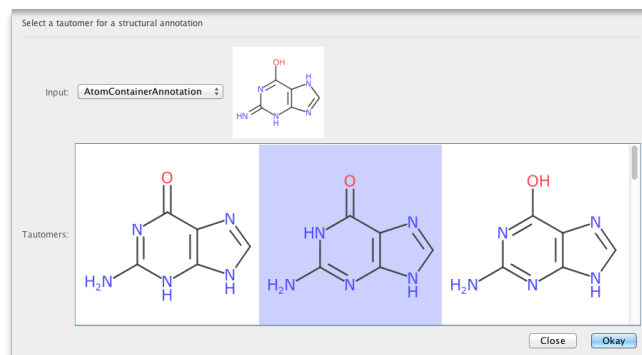


Figure 3.22: Selecting a tautomer for guanine.

alternative forms for a single metabolite and allows a user to select the preferred representation (Fig. 3.22).

Ring-chain tautomerism involves the interconversion of a structure between chain and cyclic forms. A common case in metabolic reconstructions is the representation of carbohydrates that can convert between aldehydo, pyranose, and furanose forms. The *furantor* and *pyranator* utilities attempt to match chain form carbohydrates and transform them to their respective furanose and pyranose cyclic forms. Metabolites are selected in the table and checked for a match against a SMARTS query (Apdx. B.4). Structures matching the query are listed for user confirmation (Fig. 3.23).

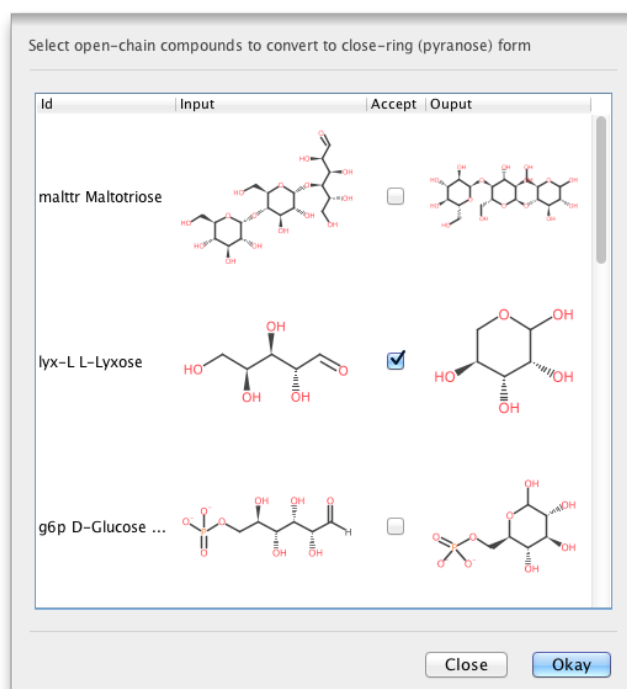


Figure 3.23: Metabolites are selected from the reconstruction and the ring-form (pyranose) listed for user confirmation.

3.4 Conclusions

A desktop application and associated library have been created to modify and manage the complex data stored in metabolic reconstructions. The underlying object model provides an appropriate representation for handling the components and supports flexible annotation types. Convenience functionality is also provided in the interface for merging, splitting, and searching (Apx. B.5).

Data sources are accessed through an abstract service framework allowing the application to operate independently. The framework can easily be adapted and extended for new data resources.

The efficient binary format provides the fast loading and storage required for an interactive interface. Export of an SBML document with MIRIAM annotations allows the sharing of reconstructions with other software.

Metabolites can be annotated with cross-references and chemical structure annotations in either an automated, semi-automated, or manual procedure. Chemical structure annotations can be created, modified, and standardised allowing representation of metabolites with specified charge and stereochemistry. The hybrid approach allows more specific annotation when an exact database identifier is not found. The chemical structure annotations are utilised to identify identical metabolites within or between reconstructions.

Metingear has been used in the manipulation and annotation of metabolic reconstructions in Chapter 2. The described methods allowed precise annotation with structures, even when a cross-reference could not be found. The import from tabular formats has identified inconsistencies (typographical errors and missing information^[146]) and highlights the need for software in creating reconstructions.

The application provides many of the steps required for building draft reconstructions but since many excellent tools already automate this functionality the focus is on manual editing and metabolite annotation. Despite this, the application has been used in the assembly of a draft network of the human skin bacterium *Anaerococcus octavius* NCTC 9810, a whole-genome sequence of which was obtained in-house by Unilever.

The core object model and application is functional but can be improved and extended. Modification of entities are tracked by the undo/redo manager but this information is not persisted or explicitly presented in the interface. Including this system

would provide versioning and make the software more useful when annotations are made by multiple users. Along a similar theme, the centralised storage of the reconstruction information and modification through a local client was suggested during peer review.

The use of a binary format to store reconstructions was primarily to allow consistent round tripping of the annotation types. Despite the overhead, using SBML as the primary storage format would simplify the library. Currently, this would require a customised schema but extensions to SBML that allow richer annotations have been proposed^[191].

A common use of chemical structure annotations is the calculation of major microspecies at physiological pH and Gibbs energy. The major microspecies is used for checking mass-charge conservation, while the Gibbs energy is used in assigning reversible reactions. A tool for calculating microspecies is JChem's pK_a toolkit^[170]. This functionality cannot be directly included due to license restrictions but it may be possible to expose the utility via an extension.

Accurate prediction of pK_a is difficult^[192] but a simple approach based on functional groups may be useful^[193]. Here, functional groups that are known to be (de)protonated at pH ~ 7 (e.g. carboxylate and organophosphate) are located and adjusted without precise pK_a prediction.

The eQuilibrator tool is a free and open source Gibbs energy predictor^[194]. Although the source code is available, the calculations again depend on dissociation constants provided by JChem.

Gene-Protein-Reaction (GRP) relationships describe which genes are required for what reactions and are essential for validating growth conditions through knock outs. Although this information is represented implicitly in the object model it cannot be easily modified. The information is only encoded if reactions are created from functional annotations or imported sequences have identifiers matching existing reaction annotations.

Metabolite annotation has focussed on exact graph representations of small molecules. Some reconstructions include generic metabolites or reactions that represent a collection of more than one discrete entity. Enumerating the generic metabolites and reactions is less compact but more descriptive. When structural annotations were added to *iAF1260* for computing atom-atom mappings, several core reactions required enu-

meration^[127]. The methods required for enumeration have been implemented and utilised by Pablo Moreno^[138] and will be included in a future release.

3.5 Availability

An executable version of Metingear is available for Windows, Macintosh OS X, and Linux at: <http://johnmay.github.io/metingear>. The webpage also links to: manuals, tutorials, and screencasts. The source code for both Metingear and the MDK is freely and openly available from the GitHub repositories:

- <http://www.github.com/johnmay/metingear>
- <http://www.github.com/johnmay/mdk>

PART II

CHEMICAL INFORMATION
PROCESSING

Chapter 4

Ring perception

4.1 Introduction

4.1.1 Motivation

A key aspect in manipulating and querying chemical information is the ability to define and reason about the properties of structures. Finding and describing the rings of a structure is a foundation for many other procedures in chemical information processing.

In representation, different resonance forms need to be treated as equivalent. One approach to this is to find rings with overlapping π -orbitals that can be delocalised. Conversely, a delocalised structure can have electrons assigned to ring systems to produce a specific Kekulé representation.

In searching, rings can be utilised in: fingerprints, backtracking isomorphism matching, and query languages. Fingerprints describe features of a structure with a set of keys, invariant ring properties can be used to describe features that are not captured by the connectivity. When matching atoms and bonds between structures, the ring properties can be used in early elimination of infeasible matches. In atom-atom mapping, they can also be used to penalise ring opening or closing bond changes. The SMARTS structure query language allows matching of ring membership, size, and number.

In stereochemistry, geometric isomers (due to double-bonds) should not be encoded

when the bond is in a rigid ring. Rigidity is approximated by only allowing stereo-configurations in rings with more than seven atoms. Rings are also needed in locating groups of intradependent stereocentres^[195].

4.1.2 Types of ring information

Considering the uses above, there are three key questions we wish to answer:

1. Is an atom or bond in a ring?
2. What size ring does an atom or bond belong?
3. What are the sets of atoms and bonds in rings (ring sets)?

The first is answered with a logical yes or no, the second with an integral value, and the third with collections of atoms and bonds. Notice that answering the second is sufficient to answer the first and answering the third is sufficient to answer the second. However, if one only needs to know whether an atom or bond is in a ring, it is much easier (and efficient) to answer this directly.

The answer to the third query depends on what we define as a “ring”. There is often a disconnect between how chemical rings are numbered and what is useful for computation. Conflicting definitions of rings contribute towards discrepancies between chemistry toolkits such as assigning aromaticity. The CDK does not enforce a single ring definition but provides multiple algorithms for different uses. Some considerations of the differences will be touched upon but a thorough review of ring sets is provided by Berger *et al*^[196] and Downs *et al*^[197,198].

Algorithms for finding rings are rooted in graph theory. An extended introduction to graph theory is provided as Appendix C.1. For this chapter, a ring is an *elementary cycle* that consists of a closed path of non-repeating (vertices) atoms and (edges) bonds. For brevity, *cycle* is used in reference to an *elementary cycle*, that in turn corresponds to the rings in a structure.

Cycle membership

Determining whether an atom or bond is in a ring (cycle membership) is important as a first step in other calculations. In PubChem-Compound^[53] (Aug 2013) 97.3% of structures (47,745,887) contained a cycle. Although high, only 59.3% of the non-hydrogen atoms and 57.3% of bonds were cyclic. Eliminating these acyclic vertices

and edges from further processing reduces the size of the computation and improves performance.

Minimum cycle basis (smallest set of smallest rings)

A well known set of cycles in cheminformatics is the Smallest Set of Smallest Rings (SSSR). The SSSR was originally defined as a minimum length Kirchhoff-fundamental basis but has evolved to refer to a minimum cycle basis (MCB, B_{\leq}). The original definition of SSSR does not always contain the shortest cycles and was computationally intractable^[196]. The term SSSR will only be used in reference to CDK implementation names in this chapter.

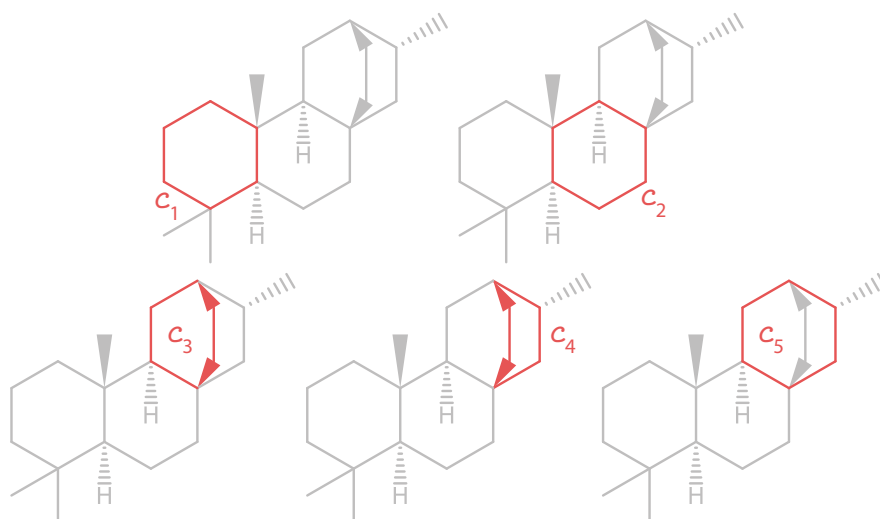


Figure 4.1: The five short cycles of CHEBI:36471. A minimum cycle basis is formed of c_1 , c_2 , and any two of c_3 , c_4 , or c_5 .

A MCB is a linearly independent set of cycles that can be used to generate the cycle space (all other cycles). A MCB is polynomial in cardinality, allowing efficient storage. From a basis, new cycles can be created by the disjoint union (\oplus -summing¹) the edges of two cycles. Notice that $c_1 \oplus c_2$ produces a new elementary cycle (not displayed in Figure 4.1).

For the example in Figure 4.1, the cycles c_3 , c_4 , and c_5 are linearly dependent. Each can be described by \oplus -summing of edges of the other two cycles. The cycle c_3 can be

¹Exclusive-or, XOR

made by \oplus -summing c_4 and c_5 .

Only four cycles are needed for a basis:

$$B_{\leq} = \{c_1, c_2, c_3, c_4\} \quad (4.1)$$

$$= \{c_1, c_2, c_3, c_5\} \quad (4.2)$$

$$= \{c_1, c_2, c_4, c_5\} \quad (4.3)$$

As there are three bases, a MCB is a non-unique set of cycles and has little direct use in similarity, aromaticity, depiction, or other descriptive features. It is also not required to find the shortest cycle through each edge or vertex which can be accomplished without checking the cycles form a basis.

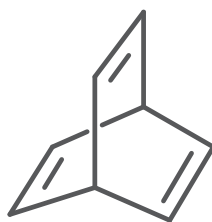


Figure 4.2: The structure of barrelene. Barrelene is thought of having two rings and mirrors what is found in a MCB and reflected in its systematic name (bicyclo[2.2.2]octa-2,5,7-triene). The choice of which two rings is arbitrary. Barrelene has three relevant cycles and no essential cycles. In this simple example the choice of which two cycles is irrelevant as they are symmetric. This would no longer be the case if structure was heterocyclic or had exocyclic group added.

Although a MCB is not unique, the number of cycles it contains is. This value is the circuit rank¹ (r) and is the number of edges that would need to be removed to make the graph acyclic (a spanning tree). For these reasons the size of an MCB agrees with de facto standards and chemical nomenclature (Fig. 4.2).

$$r = |E| - |V| + |\text{ConnComp}(G)| \quad (4.4)$$

¹alternatively known as cyclomatic number, nullity (μ), frère jacque number, first Betti's number, or bond closures^[198].

Equation 4.4 provides the circuit rank without computation of the actual cycle basis^[198]. The connected components ($ConnComp(G)$) of a graph can be found in linear time with a depth-first search^[199].

Essential and relevant cycles

The essential and relevant cycles are uniquely defined sets of short cycles. The essential cycles ($B_{<}$) is the intersect of all minimum cycle bases whilst the relevant cycles ($B_{=}$) is the union. The essential and relevant cycles for Figure 4.1 are:

$$B_{<} = \{c_1, c_2\} \quad (4.5)$$

$$B_{=} = \{c_1, c_2, c_3, c_4, c_5\} \quad (4.6)$$

When a graph has a single MCB, it is equal to both the essential and relevant cycles. As a subset of a MCB, the essential cycles do not form a basis and cannot be used to generate the cycle space. Like a MCB the essential cycles are always polynomial in number. Perhaps, counterintuitively, structures such as barrelene (Fig. 4.2) contain no essential cycles. As a superset, the relevant cycles do form a basis but may be exponential in number. The relevant cycles is the smallest uniquely defined cycle set that forms a basis.

All elementary cycles

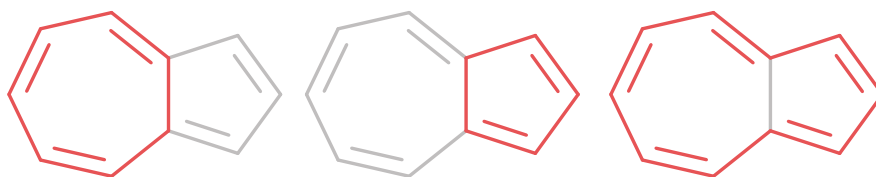


Figure 4.3: The 3 elementary cycles of azulene.

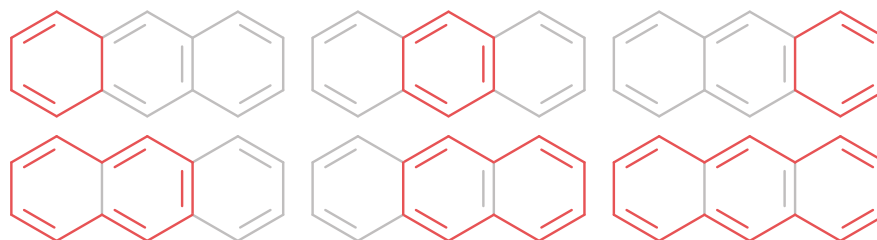


Figure 4.4: The 6 elementary cycles of anthracene.

When considering all cycles, the number can be large (Fig. 4.5) and infeasible to compute for fullerene-like and cyclophane-like structures. This set of all cycles includes envelope rings in structures like azulene (Fig. 4.3) as well as embedded rings in structures like anthracene (Fig. 4.4).

These cycles are primarily useful for aromaticity calculations but have also been utilised in scoring atom-atom mappings. The set of all cycles can be generated combinatorially using a cycle basis or computed directly through graph reduction^[200].

4.1.3 Objectives

Algorithms for computing the cycles of graphs are fundamental to processing chemical information. They are used as building blocks in many other procedures and it is essential the algorithms are both efficient and fast.

The existing implementations available in the CDK are:

- SpanningTree
- SSSRFinder
- AllRingsFinder

The SpanningTree was previously introduced in the CDK (2005) to eliminate acyclic vertices and edges, reducing the runtime of existing algorithms^[201]. It uses a greedy algorithm^[202] for weighted graphs that sequentially builds up a spanning tree (Apdx. C.2). As graphs representing chemical structures are unweighted, a more specific algorithm can be used.

The existing SSSRFinder finder provides the calculation of a MCB, the essential cycles, and the relevant cycles. The implementation uses an efficient algorithm suited to large sparse graphs^[203].

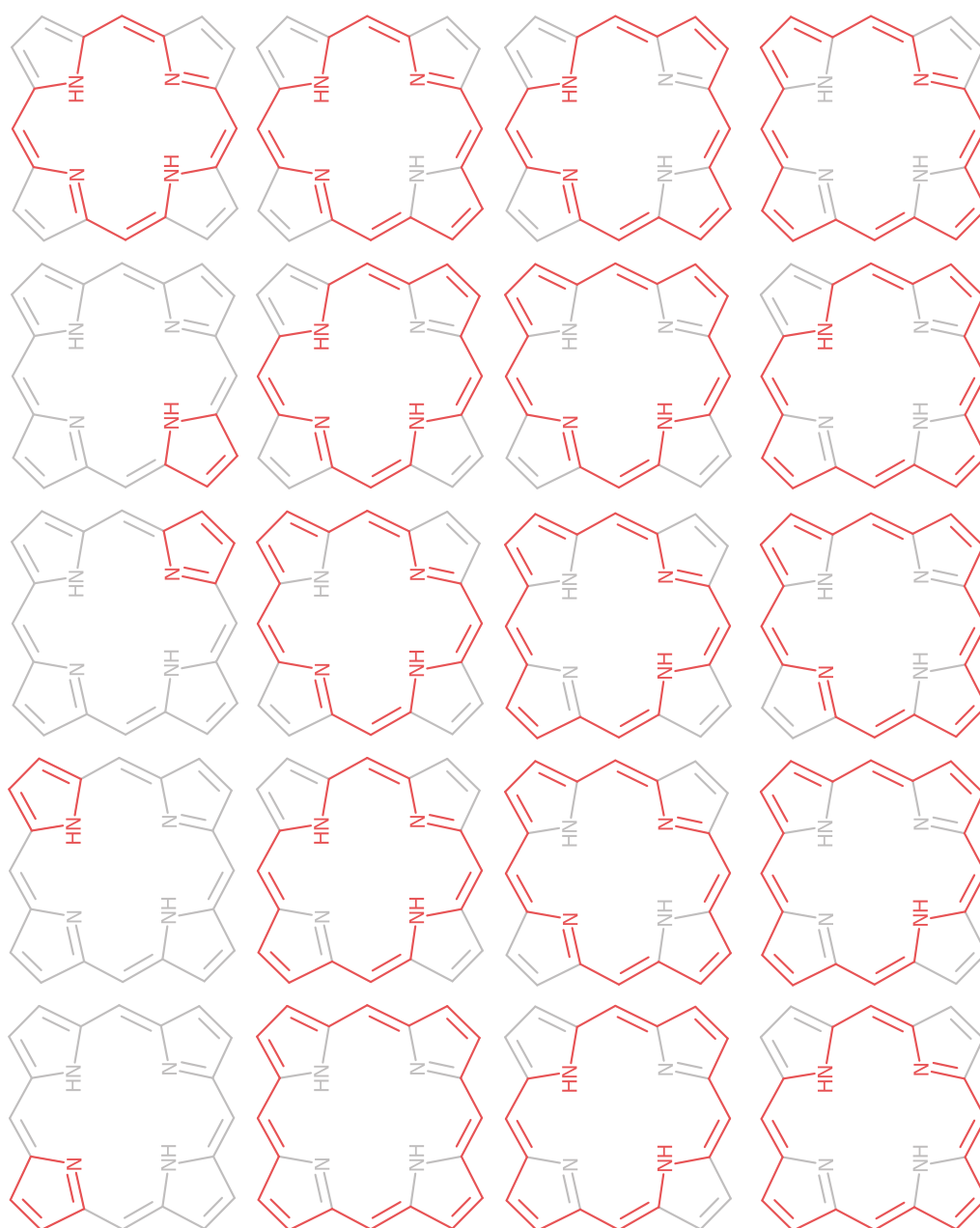


Figure 4.5: The 20 elementary cycles of porphyrin.

Throughout the CDK, a non-unique MCB is used more than the unique cycle sets. This is partly because some methods were written before the unique sets were available but is also because the more favourable unique sets (essential and relevant cycles) take much longer to compute. Providing these sets in equivalent time would encourage their use as a replacement for a non-unique MCB.

The existing AllRingsFinder computes all elementary cycles directly. As the number of elementary cycles is exponential, the algorithm is intractable for some structures. A time based threshold is used to test for infeasibility. If after a set time (5 s) the algorithm is still running, the calculation is aborted. Whether the algorithm completes depends on the hardware specifications and the current load on the CPU. For the intractable cases, a massive numbers of cycles are being generated and more computing resources are consumed. This not only slows down the execution but also means there is more memory to be freed and reclaimed on termination. Ideally, a threshold is reached before (or soon after) this happens. The arbitrary timeout does not provide this and the default time of 5 seconds actually has little gain over a smaller value (Apdx. C.3).

To address these issues, this chapter describes algorithms and optimisations for calculating cycles. The new algorithms are compared to the existing implementations as a demonstration of effectiveness.

4.2 Methods

Source code of all implementations are openly available in the CDK. Full details are listed at the end of the chapter.

4.2.1 Graph data structures

A graph can be represented and stored using several data structures^[199] (Fig. 4.6). The choice of data structure can dramatically affect performance. A coordinate or edge-list representation stores the vertices and edges as separate lists. The edge list is memory efficient but inefficient to determine adjacency where every edge must be checked. An adjacency matrix is a square matrix with a boolean value indicating whether two vertices are adjacent. Matrix representations offer constant time adjacency checking but require every vertex to be checked in order to obtain a list of neighbours or degree. The matrix representation is less memory efficient and requires quadratic space to store. In an adjacency/incidence list each vertex stores adjacent vertices or incident edges. Testing adjacency is bounded by the number of adjacent vertices, the *degree*^[199]. The *degree* and the set of adjacent vertices can be obtained in constant time.

The choice of data structure depends on properties being modelled, and which algorithms will be used. Chemical structures are generally small ($|V| < 100$) and each vertex is only adjacent to a few other vertices (sparse). Although more costly in memory and for modifications, the attributes of chemical structures make the adjacency (or incidence) list representation preferable.

The CDK currently uses an edge list representation to store chemical structures. Conversion of the CDK native data type to an adjacency list is relatively quick but can become significant if carried out multiple times. Many of the existing algorithms used an optimised representation but benefit was seen by avoiding the slower CDK native objects and minimising reconversion. The overhead introduced for converting the CDK objects (Tab. 4.1) could be minimised by loading directly into a more optimal data structure. When comparing to existing methods the conversion time is included.

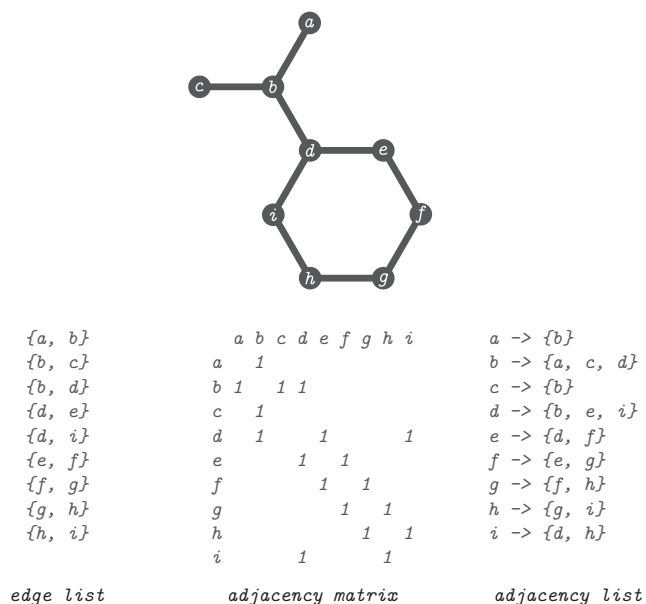


Figure 4.6: An abstract graph representation of a chemical structure (propan-2-ylbenzene) and different graph representations. The labelled vertices ($a, b, \dots i$) correspond to the values in the representations.

Table 4.1: Average ($n = 15$) time taken to convert native CDK structures to adjacency and incidence list representations optimised for cycle finding. The datasets are listed in Table 4.2.

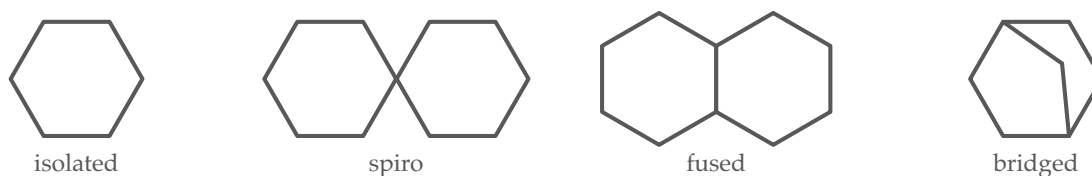
Chemical structure	n structures	Adjacency list		Incidence list	
		t (ms)	sdev	t (ms)	sdev
chebi_108	26,790	167	14	238	15
nci_aug00	250,172	998	49	1,347	53
zinc_frag	504,074	1,466	13	2,029	29
chembl_17	1,318,180	8,308	33	11,977	246
zinc_leads	5,135,179	22,537	582	33,567	2368

4.2.2 Cycle membership

The existing algorithm used in the SpanningTree implementation is for graphs with weighted edges^[202]. In an unweighted graph, any spanning tree is a minimum spanning tree. A spanning tree in an undirected, unweighted graph can be constructed with a depth- or breath-first-search^[204]. Although efficient in construction, the spanning tree still requires additional operations to determine the cyclic vertices and edges. A more efficient approach is to compute the biconnected (2-connected) components of the graph.

The biconnected components can be found using a single depth-first search^[205]. A vertex is *biconnected* if removing it from the graph does not increase the number of components. A *biconnected component* is a maximal connected subgraph where every vertex is biconnected. In addition to detecting the cyclic vertices and edges the procedure also partitions the graph into separate components which correspond to separate ring systems in the chemical structure.

If the number of edges in a biconnected component is equal to the number of vertices, $|E| = |V|$, then the circuit rank (r) is 1. These components are a single elementary cycle; they shall be referred to as *simple*. Simple biconnected components are found in isolated and spiro¹ rings whilst the non-simple biconnected components are the fused and bridged ring systems:



The simple biconnected components are edge disjoint with all other cycles and must be linearly independent. They are always a member of any basis and do not contain any other cycles ($r = 1$). This means they can be ignored from the more intensive algorithms (introduced next) and simply appended to a calculated cycle set.

The new RingSearch utility provides this functionality with an algorithm optimised for small graphs using binary sets.

¹For the example depicted here, there are two simple biconnected components for the spiro ring.

4.2.3 Minimum cycle basis, essential and relevant cycles

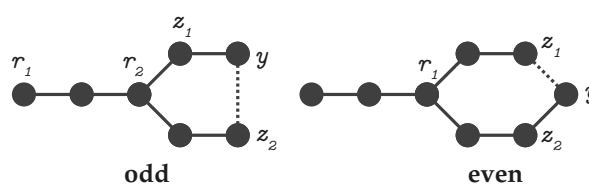
The computation of relevant cycles first computes a set of initial cycles^[206]. A MCB and the essential cycles can also be derived from this initial set.

Shortest paths

Unsurprisingly, these short cycle sets require calculation of shortest paths. A key implementation requirement for the relevant cycles is that all shortest paths of equal length can be located and efficiently stored. There may be exponentially many shortest paths between two points in a graph. Using dynamic programming, the paths can be stored and generated when required. Shortest paths are discovered with a breath-first-search from a start vertex. Normally when a vertex is found to be equal distance from the root (start), it is skipped. To find all shortest paths, a pointer is appended as an alternative branching route. From an end point, all paths can be generated by systematically exploring all branch points. The implementation in CDK that provides this is ShortestPaths.

Initial cycles

The initial cycles are found by calculating and storing multiple shortest path searches. Odd and even cycles are discovered separately.

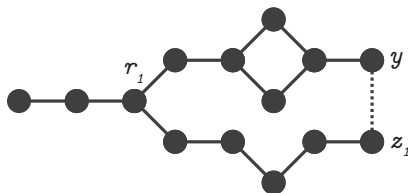


From each vertex r , a shortest path search is started. For each reachable vertex (y) adjacent vertices (z) are tested for non-interesting paths to r .

When discovering odd cycles, the path from $r_2 \rightarrow z_1$ and $r_2 \rightarrow y$ are tested, tracing back the paths we see the $r_2 \rightarrow z_1$ is contained in $r_2 \rightarrow y$ and the paths are discarded. However, the paths $r_2 \rightarrow z_2$ and $r_2 \rightarrow y$ are disjoint and an odd cycle is discovered. Note that the paths $r_1 \rightarrow z_2$ and $r_1 \rightarrow y$ are not disjoint and are also discarded. For odd cycles, only vertices that are of equal distance from the root need testing. In reality the paths to z_1 and y do not need to be checked.

The process is similar for even cycles except that any adjacent vertex may find the cycle. Here, both $r_1 \rightarrow z_1$ and $r_1 \rightarrow z_2$ are disjoint with the path $r_1 \rightarrow y$.

As noted, all shortest paths are found and stored when there is a branch point. For this example here $r_1 \rightarrow y$ has two alternate shortest paths.



To avoid discovering the same cycle multiple types, an ordering is placed on the vertices^[207]. Paths are only allowed if all vertices in the path are descending (e.g. $r_1 \rightarrow y$ where $r_1 > y$). To ensure a polynomial number of shortest paths searches, the vertices are ordered by degree^[206].

With the vertices ordered by degree if the biconnected component is known to be a non-simple component (fused or bridged) then only shortest path searches from vertices with $\text{deg} > 2$ will yield new cycles. In practice this optimisation significantly reduces the number of shortest path searches in chemical structures. For the common naphthalene and anthracene fused rings, the number of searches is reduced from 10 and 14 to 2 and 4.

The set of initial cycles are discovered with the InitialCycles implementation.

Basis formation

From the initial set of cycles, a basis is formed by incrementally adding candidate (initial) cycles of increasing size. A candidate is added to a basis if it is linearly independent from the current members^[207]. This check for linear independence is expensive¹ and can be avoided under some conditions. With the union of all edges in a basis (E_B) a new cycle is linearly independent if any edges of the candidate (E_{cand}) are not present in basis. That is, when $|E_B \cap E_{cand}| < |E_{cand}|$, the cycle must be independent. The BitMatrix provides this linear independence check.

In a minimum cycle basis, a cycle is added if it is linearly independent of all cycles of equal or smaller size (B_{\leq}). For the relevant cycles, the independence is only checked

¹linear independence is checked with row reduction of a matrix (Gaussian elimination).

with cycles of smaller size ($B_{<}$). For the essential cycles, the set is formed in a leave one out manner. A cycle is only added if there is no other cycle of equal size ($B_{=}$) that can replace it.

An additional optimisation for finding a MCB is to finish building a basis when the number of cycles equals the circuit rank. As the biconnected components are processed separately, the circuit rank of the component is $|E| - |V| + 1$.

4.2.4 All elementary cycles

The data structures of AllRingsFinder were optimised and the timeout replaced with a threshold specific to the algorithm^[200]. The improvements to the data structures involved representing the path-graph as an incidence-list and using binary sets to test intersection.

The algorithm progresses by iteratively reducing (removing) vertices¹ – the order of removal can be predetermined or dynamic. Using a predetermined order, the edges only need to be indexed by the next endpoint (i.e. directed).

This significantly reduces the number of modifications to the path-graph. Edges are only removed when a vertex is being reduced and all edges can be removed at once from this vertex. This means that edges are only ever appended to each adjacency list and individual items are never removed.

As each vertex is reduced, the degree on the adjacent vertices may increase. Limiting this degree provides a threshold to determine infeasibility. If the number of edges connected to a vertex exceeds a specified limit, the calculation can be aborted. A meaningful choice for this degree limit is explored in the results.

¹An animation is available at <http://efficientbits.blogspot.co.uk/2013/06/implementing-hansers-path-graph.html>

4.3 Results and discussion

To measure the performance of different implementations, five publicly available chemical datasets were used (Tab. 4.2). Due to difference in size, performance is listed as structures per second. The absolute times taken are available in Appendix C. All run-times were measured on an Intel Core i7 CPU (2.66 GHz) with 4 GB of RAM.

Table 4.2: Chemical structure sets used to measure performance. The number of structures is the number which were successfully read from SMILES^[101] notation. The ChEBI and ChEMBL datasets had a small number of erroneous SMILES string which could not be interpreted.

Identifier	<i>n</i> structures	Description	Available
chebi_108	26,790	ChEBI Release 108 ^[44]	www.ebi.ac.uk/chebi
nci_aug00	250,172	NCI Aug 2000 ^[208]	cactus.nci.nih.gov/download/nci
zinc_frag	504,074	Zinc Clean Fragments Ph7 2013-04-12 ^[209]	zinc.docking.org/subsets/clean-fragments
chembl_17	1,318,180	ChEMBL Release 17 ^[210]	www.ebi.ac.uk/chembl
zinc_leads	5,135,179	Zinc Clean Leads Ph7 2013-05-31 ^[209]	zinc.docking.org/subsets/clean-leads

4.3.1 Cycle membership

The time taken to determine cycle membership was measured for the new RingSearch and the existing SpanningTree (Fig. 4.7). The new algorithm can process between 100,000 and 300,000 structures per second. The majority of time taken (Apdx. C.4.1) for RingSearch was actually in the conversion to the adjacency list (Tab. 4.1).

The zinc_leads dataset has almost five times the number of structures than chembl_17 but the SpanningTree actually finished in less time (Apdx. C.4.1). This is because chembl_17 contains more fused-ring systems that are problematic for the SpanningTree. The difference is emphasised by measuring the performance on single structures containing large fused ring systems (Tab. 4.3).

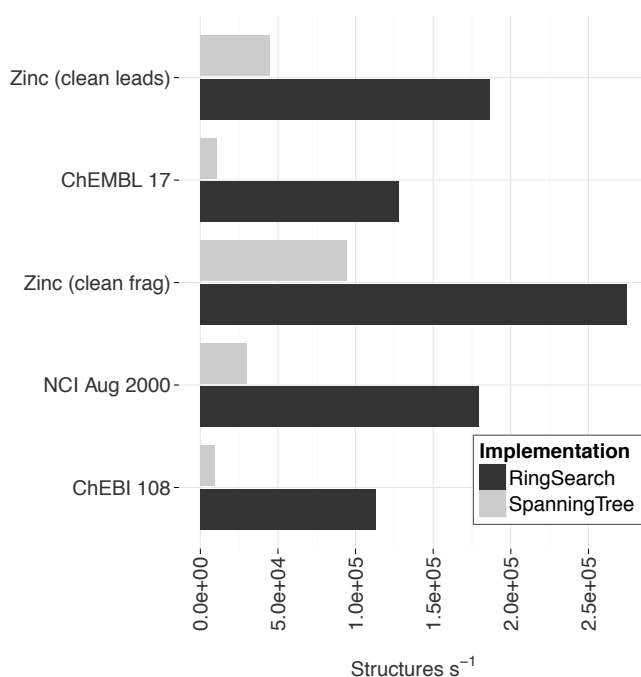


Figure 4.7: Improved cycle membership perception. The performance (structures per second) when using the RingSearch compared to the pre-existing SpanningTree. Times measured for the RingSearch include the conversion from the CDK objects to an *adjacency* list representation (Tab. 4.1).

Table 4.3: Average ($n=50$) time taken to determine cycle membership in structures with fused-rings from ChEBI and FULLERENE^[211]

Chemical structure	SpanningTree		RingSearch	
	t (ms)	sdev	t (ms)	sdev
cubane (CHEBI:33014)	1.18	1.24	0.21	0.66
dodecaboride (CHEBI:51706)	1.11	0.80	0.05	0.01
octacontaboron (CHEBI:50252)	100.18	52.87	0.44	0.18
C ₆₀ fullerene (CHEBI:33128)	11.15	2.92	0.30	0.03
C ₇₀ fullerene (CHEBI:33195)	10.44	4.25	0.88	0.23
C ₃₂₀ fullerene (FULLERENE)	423.06	166.30	0.61	0.32
C ₇₂₀ fullerene (FULLERENE)	3100.71	1171.62	1.65	0.66

The RingSearch implementation indicates whether an atom or bond is cyclic and whether it is in a simple biconnected component. Simple biconnected components are edge disjoint with all other cycles and can be removed from further processing.

To measure the impact of removing the simple components, the time taken to compute a MCB was measured with different levels of preprocessing (Fig. 4.8). Although processing only the cyclic vertices and edges improves performance, an even larger gain can be seen by only processing the non-simple biconnected. The largest performance improvement was seen for the zinc datasets that contain fewer fused ring systems.

4.3.2 Minimum cycle basis, essential, and relevant cycles

The performance of the new MinimumCycleBasis, RelevantCycles, and EssentialCycles was compared to the existing SSSRFinder. The performance measurements of the existing implementation on zinc_leads dataset was not stable and are not reported.

Although the new MCB algorithm has a higher computational complexity than the existing SSSRFinder, in practice it was found to be several factors faster (Fig. 4.9). This is because the efficiency describes how the algorithm scales based on the number of vertices (atoms) and edges (bonds). Graphs representing the chemical structure of metabolites typically small and sparse.

The time taken for processing zinc_frag has dropped from ~ 47 to ~ 2 seconds whilst processing chembl_17 went from ~ 4 minutes to ~ 15 seconds (Apdx. C.4.2).

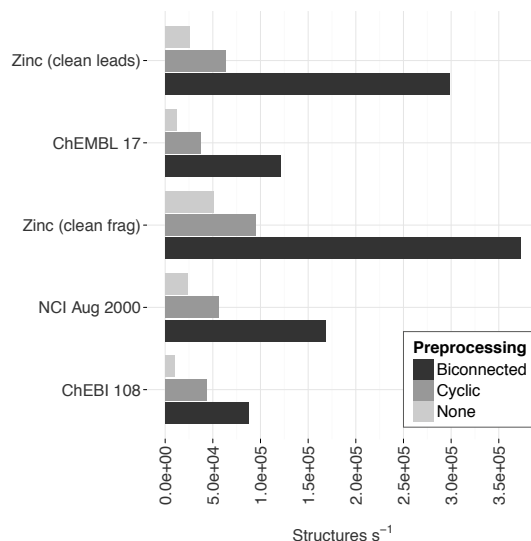


Figure 4.8: The impact of varied pre-processing when computing a MCB. *none* indicates that a MCB was computed on the entire graph whilst *cyclic* indicates that a MCB was computed on a subgraph of only the cyclic vertices and edges. The *biconnected* pre-processing computed a MCB only on the non-simple biconnected components. The performance includes both the application of the pre-processing (i.e. finding the biconnected components) and the computation of a MCB. The time taken to convert the CDK objects (Tab. 4.1) is not included.

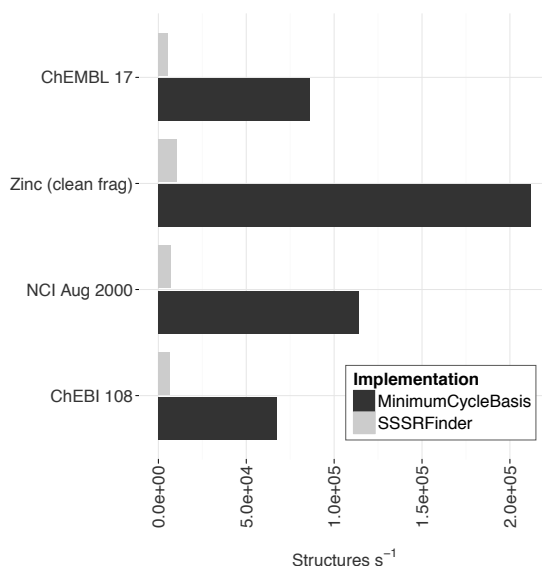


Figure 4.9: Performance of computing minimum cycle basis. A comparison of computing a Minimum Cycle Basis (MCB) using the existing (SSSRFinder) compared to the new MinimumCycleBasis. The performance includes conversion to an *adjacency* list representation for the new method (Tab. 4.1).

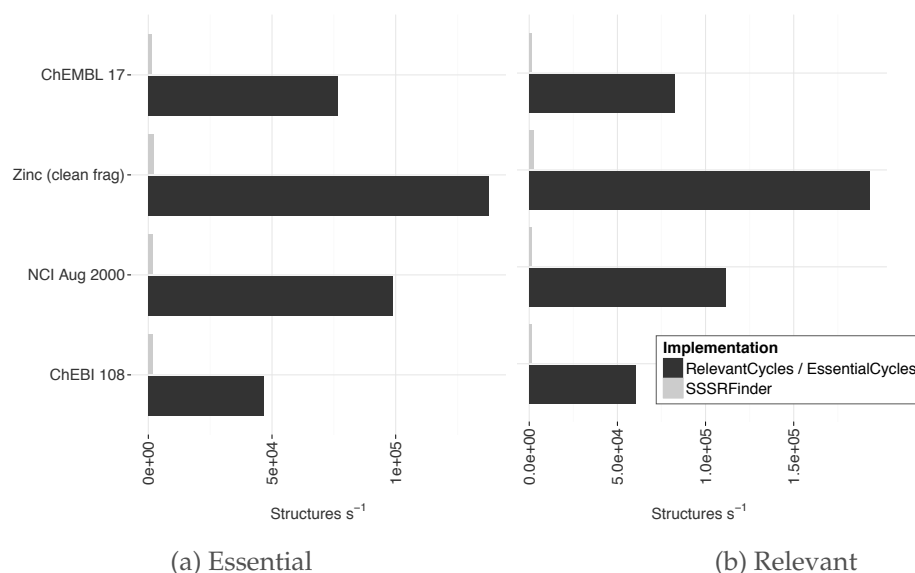


Figure 4.10: Performance of computing relevant cycle basis. A comparison of computing the essential and relevant cycles using the existing (SSSRFinder) and new implementations. The performance includes conversion to an *adjacency* list representation for the new method (Tab. 4.1).

The relative performance improvement of the relevant and essential cycles was larger than for a MCB (Fig. 4.10). The existing implementations could process around 2,000 structures per second whilst the newer implementations can process between 50,000 and 200,000. The time taken to find all the relevant cycles in chembl_17 previously took ~16 minutes and now takes only ~16 seconds (8 seconds of which are spent in conversion) (Apdx. C.4.3). Similarly, the time taken to find the essential cycles in the nci_aug00 dataset went from ~2.5 minutes to ~2 seconds (Apdx. C.4.4).

Table 4.4: The number of cycles found in chemical datasets for non-unique and unique cycle sets. The counts for all elementary cycles are approximate as it was infeasible to finish computation on some structures.

Chemical structure	<i>n</i> structures	MCB	Essential	Relevant	All
chebi_108	26,790	56,572	55,687	57,401	~126,713
nci_aug00	250,172	599,876	591,144	606,045	~1,007,643
zinc_frag	504,074	880,296	875,801	882,393	~1,022,498
chembl_17	1,318,180	4,505,285	4,455,907	4,563,027	~6,599,942
zinc_leads	5,135,179	-	-	-	~14,816,752

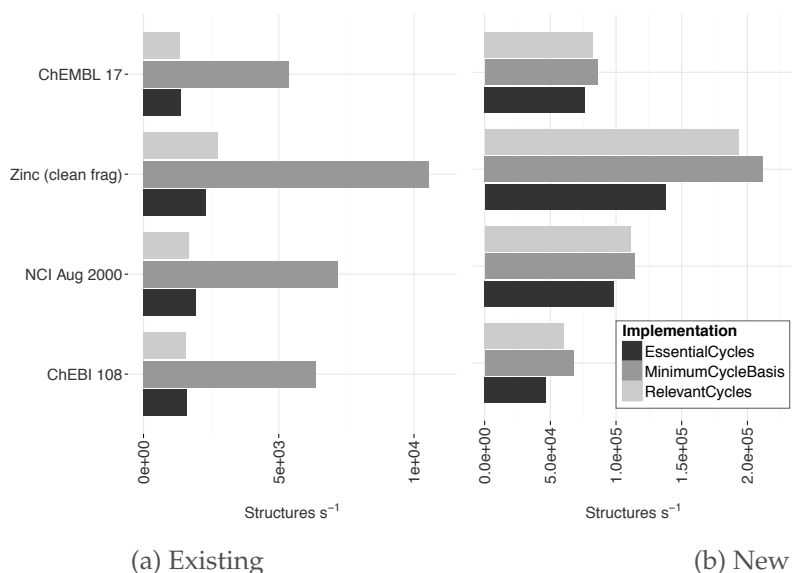


Figure 4.11: Performance comparison of the old MCB, Essential and Relevant cycles. Previously the computation of a MCB is much faster than the essential and relevant cycles. This led to a MCB being favoured for use in other procedures.

A key objective was to provide the calculation of unique relevant and essential cycles in equivalent time to the non-unique MCB. The relative difference between the unique sets and a MCB was substantially reduced (Fig. 4.11). The calculation of the essential and relevant cycles is still slightly slower because checking the initial cycles is more demanding than for a MCB.

For the essential and relevant cycles to be used in place of the non-unique MCB the number of cycles should also be similar. The number of cycles found by each algorithm was measured on the datasets (Tab. 4.4).

As expected there were less essential cycles and more relevant cycles compared to a

MCB. On average the number of cycles found in the unique sets was within 1% of the non-unique MCB. In practice this means that the unique sets can be used as a replacement for the non-unique MCB with minimal impact.

It is inevitable that some structures will have an infeasible number of relevant cycles. During testing, a structure in PubChem-Compound ([CID 53389303](#)) was found to contain over 1 million relevant cycles. As the number of relevant cycles can be determined without generating the paths, such structures can be filtered out if desired.

4.3.3 All elementary cycles

The new AllCycles implementation requires a threshold value to determine infeasibility. To determine an appropriate threshold the algorithm was run on all fused ring systems found in PubChem-Compound (Dec 2012). The maximum degree required to perceive the ring systems was measured and reported for every structure. An arbitrarily high value of ($deg = 5000$) was chosen as an absolute maximum. There were 987 (0.005%) ring systems that would require a threshold larger than this to finish.

As the required thresholds were recorded, percentiles could be retrospectively calculated. Only a small gain is seen for higher values (Tab. 4.5) and a threshold of only 9 allows perception of 99% of the ring systems. The default value (used in benchmarks) was chosen as 684 (~99.99%). The threshold parameter is now selected by choosing one of the precomputed percentiles.

Table 4.5: Required threshold to compute the given number of ring systems which it would be feasible to find all cycles

Percentile	Threshold (<i>degree</i>)	Feasible ring systems	Infeasible ring systems
99.95	72	17,834,013	8,835
99.96	84	17,835,876	6,972
99.97	126	17,837,692	5,156
99.98	216	17,839,293	3,555
99.99	684 (default)	17,841,065	1,783
99.991	882	17,841,342	1,506
99.992	1,062	17,841,429	1,419
99.993	1,440	17,841,602	1,246
99.994	3,072	17,841,789	1,059
99.9946	5,000 (max tested)	17,841,861	987

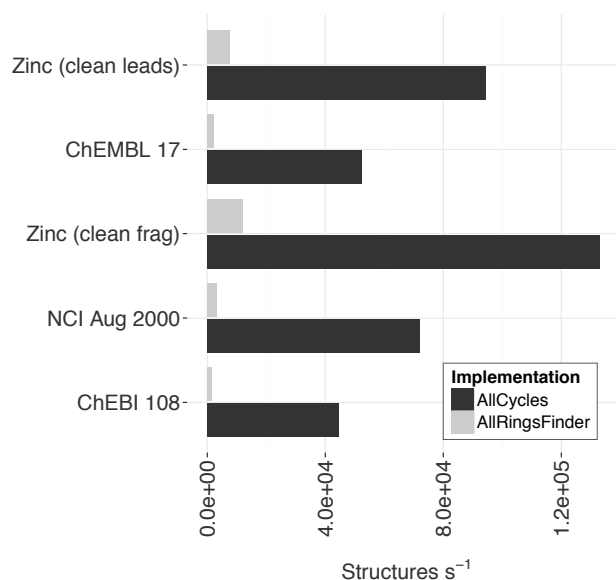


Figure 4.12: Performance comparison when finding all elementary cycles. The performance difference between the existing and new implementations for finding all elementary cycles. The performance includes conversion to an *incidence* list representation (Tab. 4.1).

The performance of the new implementation was compared against the existing algorithm. A small time out of 5 ms (much lower than default) was chosen for the existing AllRingsFinder.

The new algorithm was able to find more cycles (Apdx. C.5.1) in less time (Fig. 4.12). The absolute time taken to find all cycles in chembl_17 previously took ~9 minutes but now takes only ~25 seconds (Apdx. C.4.5). Perhaps most surprising is the performance is comparable to the new unique cycle sets and MCB¹. Even though many more cycles are located (Tab. 4.4).

¹Note that times reported for the AllCycles performance were measured with conversion to the *incidence* and not *adjacency* graph data structures use for the short cycles.

4.4 Conclusions

Efficient and fast algorithms for ring perception have been implemented in the CDK. The performance has been compared on several datasets and shows dramatic improvements. The improved performance means it is now feasible to analyse much larger chemical datasets and in less time.

Ring perception algorithms are used as building blocks for other procedures and improvements were utilised elsewhere in the toolkit. The new algorithm for cycle membership has been used in the hash codes (Chap. 2) and to improve performance of atom typing.

Each algorithm described in the chapter can be invoked separately or through a new Cycles facade (Listing 4.1). The facade is a high-level API that provides simplified access to algorithms and takes care of conversion and optimisations. It also provides utilities to combine and filter cycle sets.

Listing 4.1: Using the Cycles facade

```
IAtomContainer ac = ...; // native CDK chemical structure class

Cycles.all(ac);           // all elementary cycles
Cycles.all(ac, 10);       // all elementary cycles |V| <= 10
Cycles.relevant(ac);      // relevant cycles
Cycles.essential(ac);     // essential cycles

// find all cycles, or if infeasible, find relevant cycles
CycleFinder cf = Cycles.or(Cycles.all(), Cycles.relevant());
cf.find(ac);
```

Using the biconnected components, the new RingSearch provides a very fast method of determining which atoms and bonds are in a ring. The biconnected components are partitioned as components that are edge disjoint with all other components need no further processing. This was demonstrated to improve the performance of other cycle sets.

The unique short cycle sets (essential and relevant) saw an order of magnitude improvement, it is no longer favourable to utilise a non-unique MCB for performance reasons. Any procedures incorrectly relying on a MCB to be unique can be easily adapted to use the new algorithms. The efficient implementation of the relevant cycles could also be used to compute the Unique Ring Families descriptor^[212].

Finding all elementary cycles is now very fast and enables improved aromaticity perception (Chap. 5). If computation is not feasible the aromaticity perception falls back to a smaller more feasible cycle set. Alternatively the smaller set of cycles could be tested first with the larger set only utilised if potentially aromatic atoms were remaining. Using the optimised data structures, the set of all cycles is generally faster to compute than the smaller sets and it is preferable to try all and fail fast. This is possible because the new threshold provides an effective means to abort.

In addition to those discussed in this chapter, other cycle sets were explored.

The CACTVS^[108] Substructure Keys utilise a set of cycles referred to as the ESSSR (not to be confused with the ESSR^{1 [198]}). These cycles are the shortest through a vertex triple $\{u, v, w\}$ and allows generation of cycles for envelope rings such as naphthalene or azulene whilst avoiding larger fused rings.

The shortest cycle through each vertex and edge are slightly different from a MCB, essential, or relevant cycles. These cycles do not always form a basis but may still be exponential. The edge short cycles mimic that used in the Open Babel^[111] toolkit.

A significant portion of the time is spent converting the CDK objects to optimised data structures. Without the conversion, the runtime performance is much slower. Further gains could be made by optimising the native data structures and removing the need for this conversion.

4.5 Availability

All implementations described in this chapter are freely and openly available from the CDK GitHub repository, <http://github.com/cdk/cdk>. The relevant Java classes referenced in the text are:

- `org.openscience.cdk.ringsearch.RingSearch`
- `org.openscience.cdk.graph.ShortestPaths`
- `org.openscience.cdk.graph.InitialCycles`
- `org.openscience.cdk.graph.BitMatrix`
- `org.openscience.cdk.graph.MinimumCycleBasis`
- `org.openscience.cdk.graph.RelevantCycles`
- `org.openscience.cdk.graph.EssentialCycles`
- `org.openscience.cdk.graph.AllCycles`

¹ESSSR: Extended Smallest Set of Smallest Rings and ESSR: Extended Set of Smallest Rings

-
- `org.openscience.cdk.graph.TripletShortCycles` - for CACTVS Keys
 - `org.openscience.cdk.graph.EdgeShortCycles`
 - `org.openscience.cdk.graph.VertexShortCycles`
 - `org.openscience.cdk.graph.Cycles` – facade API

Chapter 5

Aromaticity, Kekulisation, and tautomerism

5.1 Introduction

5.1.1 Motivation

The use of Lewis structures to store and manipulate chemical structures provides a static view of electron location. In reality, all compounds exhibit electron mobility and many bonds are not localised. A direct consequence of this is multiple valid representations for the same compound. To effectively reason about the equivalence of chemical structures, it is essential that one can accurately assign and move electrons.

For simple structures, it is feasible to store and depict multiple resonance forms (Fig. 5.1). However, the combinatorial nature of enumerating all equivalent representations means the approach is impractical for larger structures. To represent the multiple equivalent forms a delocalised *aromatic* bond may be used. The delocalised bond is treated as though it were both a single (σ) and double (π) bond.

A bond may be indicated as delocalised by either assigning an auxiliary property to an existing single or double bond or by using an explicit delocalised bond type.

An explicit bond type is utilised in several file formats, most notably SMILES and molfile. SMILES was designed to be *canonicalisable* and provide a unique representation^[83]. Without the use of an aromatic bond type, two different SMILES for o-xylene

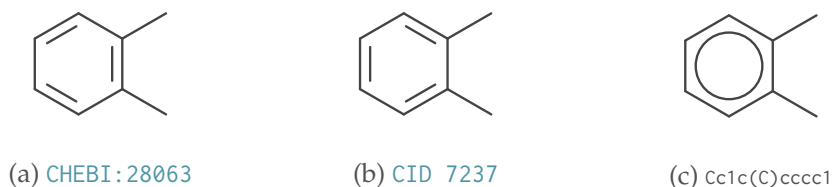


Figure 5.1: Kekulé forms of o-xylene, in ChEBI (5.1a) and PubChem-Compound (5.1b), are encoded as different SMILES: CC1=C(C)C=CC=C1 and CC1C(C)=CC=CC=1. When lower case symbols are used to represent aromaticity, they have the same encoding: Cc1c(C)cccc1.

are encoded (Fig. 5.1). The CTfile (molfile) and CML Molecule Convention also define an aromatic bond type. In the CTfile, it is a query feature^[97] and should therefore not be used to store a delocalised representation. Unfortunately entries that utilise the query *aromatic* bond are often found. The CML aromatic (“A”) bond order can be used when the hydrogen count is specified. Alternatively, an extension to CML allows the aromatic property to be auxiliary to the bond order. The InChI does not use an aromatic bond but also does not store any bond order information^[163].

To be able to effectively handle structures from different formats it must be possible to convert between the representations. This requires fundamental methods for both delocalising and localising structures. As a structure may be input or output in different forms, fast implementations allow conversions to be applied automatically.

5.1.2 Aromaticity

The concept of aromaticity is well established but somewhat ill-defined and used for several different concepts. Other than smelling nice, it is used for both describing why certain compounds are more stable than expected and for representing delocalised cyclic systems. A consequence of this is the existence of multiple aromaticity models between and within cheminformatics software^[213].

Algorithms for assigning aromaticity follow a relaxed form of Hückel’s rule^[214,215] being applied to both isolated and fused rings. The rule predicts a ring as aromatic when it is planar and the sum of p-orbitals is equal to $4n + 2$ ^[216]. The planarity check is usually relaxed to a per-atom basis that only checks orbital hybridisation.

In the existing CDK (1.4) there are three different implementations for detecting aromaticity. The main implementation used throughout the library is the `CDKHuckel-AromaticityDetector`. The other implementations are: a recent extension that allowed

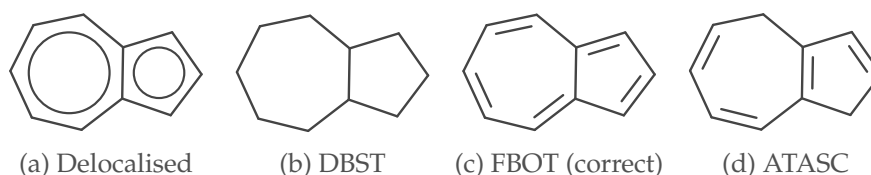


Figure 5.2: Assignment of double bond locations to azulene using existing CDK algorithms. Abbreviations are DBST: DeduceBondSystemTool, FBOT: FixBondOrdersTool, and ATASC: AtomTypeAwareSaturationChecker

exocyclic bonds and a legacy method that only tested provided rings.

The `CDKHuckelAromaticityDetector` uses information from assigned atom types to count electrons in the non-unique smallest set of smallest rings (SSSR). To detect delocalised bonds in structures such as azulene, all rings in small cyclic systems are also checked.

5.1.3 Kekulisation

The conversion of a delocalised representation to one with alternating double and single bonds is known as Kekulisation or dearomatisation.

The existing CDK library provides several algorithms that attempt to place electrons through a variety of methods. The `DeAromatizationTool` is capable of assigning bonds to benzene, pyridine, and pyrrole rings. The rings are not identified automatically and need to be provided to the method. Similarly, the `DeduceBondSystemTool` (DBST) attempts to assign tentative bond orders to five, six, and seven member rings. All possible combinations of the decisions are then considered as a whole on the molecule and the best solution is returned. The best solution may still be incorrect. The `FixBondOrdersTool` (FBOT) utilises a non-unique smallest set of smallest rings, first assigning bond orders for which there is no choice and then randomly choosing assignment thereafter. Atoms that have no double bond assigned are reported but not corrected. The `AtomTypeAwareSaturationChecker` (ATASC) attempts to find a solution that saturates all cyclic atoms using matrix operations. If a solution is not found a best guess is provided.

A common theme of all these methods is the dependence on ring detection. As there is no way of knowing which rings were used in the assignment of the delocalised bonds, a solution must check all assignments to the potentially exponential number

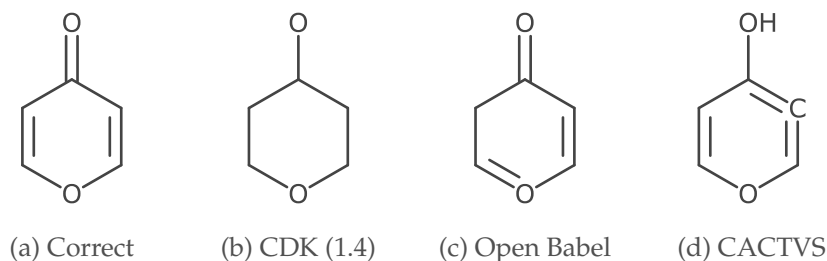


Figure 5.3: Interpretation of the SMILES oc1ccocc1 for pyran-4-one. The inclusion of an acyclic aromatic bond can lead to incorrect interpretation. The CDK (1.4) performs no Kekulisation on input and RDKit (not shown) rejects the compound as erroneous. The correct interpretation is obtained from Daylight, MarvinSketch, ChemSketch, Accelrys Draw, and CDK (1.5).

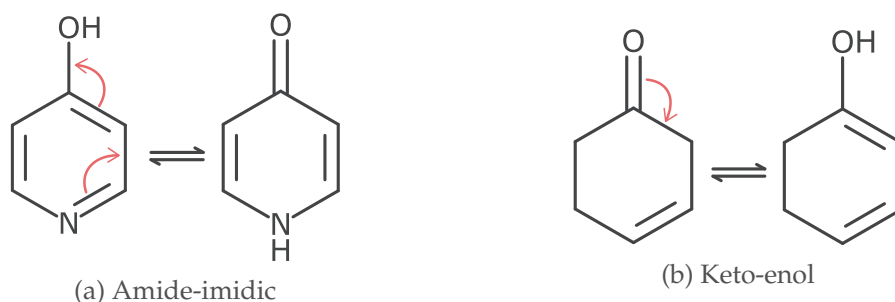


Figure 5.4: Tautomerism due the relocation of a hydrogen between different elements. In these examples the hydrogen moves across 5 (a 1,5-shift) (5.4a) and 3 (a 1,3-shift) (5.4b) atoms.

of all rings. In azulene for example, it is the 10 member envelope ring that requires alternating bonds and not the smaller rings (Fig. 5.2). Although the FBOT provides the correct assignment for azulene (Fig. 5.2c), it does not find the correct solution for other structures (e.g. porphyrin).

The use of rings in Kekulisation is not unique to the CDK library. Although perverse, the SMILES cc and ccc can correctly be interpreted as C=C (ethene) and C=CC=C (but-1,3-diene). Such compounds are unlikely to be encountered in datasets but may be entered manually. Support for non-ring delocalised bonds is varied (Fig. 5.3). Although the use of non-ring aromatic bonds is rare, a general approach that does not depend on ring perception is still beneficial.

5.1.4 Tautomerism

Tautomerism due to hydrogen mobility is a different problem to that of aromaticity or Kekulisation (Fig. 5.4). However, the algorithms to transform structures are related

and depend on manipulating alternating bonds^[217]. The relocation of a hydrogen may be accomplished with either a local or global algorithm^[218]. The local algorithms look for known patterns and rules to be applied^[219] whilst the global approaches attempt to mimic the chemistry of tautomerism^[220]. The rule-based algorithms are extendable and allow encoding of specific and possible rare shifts whilst the global algorithms may identify shifts not considered by the local rules.

The existing CDK library provides the ability to enumerate tautomers by interrupting a generated InChI^[221]. The algorithm does not provide a unique tautomer or scoring. The InChI algorithm is based on local rules and by default only identifies hydrogen atoms that move up to three atoms (1,3-shift). The InChI can also identify 1,5-shifts and keto-enol by setting additional non-standard options (15T and KET). Note that the default InChI options do allow 1,5-shifts in rings.

AMBIT is a collection of software modules for toxicology prediction based on the CDK codebase. The ambit-tautomer module provides a comprehensive set of local rule definitions for handling different types of tautomerism^[222]. Each rule is encoded with an energy and the rules handle many different types of tautomerism. The rules are incrementally applied to enumerate all tautomers. A score is assigned based on the energy of each rule and the aromaticity characteristics. The lowest scoring tautomer is considered the unique tautomer.

5.1.5 Objectives

The CDK aromaticity implementation is sufficient and well tested but the introduction of faster ring detection algorithms in the previous chapter (Chap. 4) allows for a faster and more modular approach.

The existing Kekulisation methods available in the CDK are insufficient in correctly assigning bond orders to many structures. The performance also meant they are not applied automatically. This in turn leads to several bugs where an aromatic bond is left with unset bond order. Several other algorithms (e.g. InChI generation) do not run unless the input is a Kekulé form.

Similar to Kekulé forms, the nature of the problem means the enumeration of all tautomers can be infeasible for even small compounds. A more pragmatic approach is to generate a single unique tautomer. Applying the existing methods to enumerate tautomers is several datasets was found to be impractical. AMBIT's tautomer generation

does provide a timeout eliminating the potentially exponential runtime but inspection of the scoring showed the unique tautomer to be inconsistent. The reported energy value is the value to transform the input into the output and not the absolute lowest energy. To then identify a truly unique tautomer, the transformations for all-vs-all must be compared.

To address these objectives, this chapter describes algorithms for assigning aromaticity, Kekulisation, and assigning a unique tautomer. To distinguish the existing and new algorithms the versions are referred to as 1.4 (existing) and 1.5 (new).

5.2 Methods

All algorithms are free and openly available, links to the source code are listed at the end of the chapter.

5.2.1 Aromaticity

As introduced, aromaticity algorithms follow a simplified form of Hückel's rule. An aromaticity model can be described by two parameters: how many p-electrons an atom contributes, and to what cycles (rings) in the graph $4n + 2$ is tested. A new implementation was created that utilised these two options to provide a modular implementation (Listing 5.1).

Listing 5.1: Creating and applying an aromaticity model

```
Aromaticity arom = new Aromaticity(ElectronDontation.cdk(),  
                                   Cycles.all());  
  
// apply the aromaticity model to a structure  
arom.apply(structure);
```

The implementation of efficient cycle perception was discussed in the previous chapter (Chap. 4). It is preferable to test all cycles in the graph to delocalise structures such as azulene and other corner cases^[223]. The existing implementation only tested all cycles if the minimum cycle basis contained fewer than four cycles. This constraint was in place for performance but the improvements in cycle perception meant it was no longer required. For structures where computing all cycles is intractable, a different set can be provided to fallback and use.

The electron donation parameter provides a method to determine the number of electrons each atom can donate. The contribution from each atom was previously computed on demand but it is more efficient to calculate all contributions at once. Currently, atoms are allowed to contribute: two, one, or zero electrons. A negative value is also used to indicate non-aromatic atoms. A summary of predefined implementations is provided in Table 5.1.

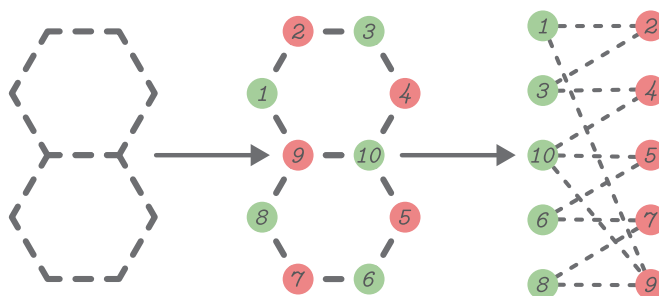


Figure 5.5: Rearrangement of the vertices in a naphthalene skeleton into two groups. The groups are displayed as a bipartite graph. Atoms (vertices) are numbered according to IUPAC nomenclature.

Table 5.1: Differences between electron donation models available in CDK (1.5).

Implementation	Double-bonds	Lone-pairs	Exocyclic bonds	Ketone contribution	Method
CDK	Yes	Yes	No	-	Preassigned atom types
CDK (exocyclic)	Yes	Yes	Yes	1	Preassigned atom types
Daylight	Yes	Yes	Yes	0	Electron counting heuristics
π -bonds	Yes	No	Yes	-	Conditional

5.2.2 Graph matching

A *graph matching* or *independent edge set* is a pairing of adjacent vertices where each vertex is paired with 0 or 1 other vertices. A vertex paired with another is matched whilst those that are not paired are unmatched. A *maximum matching* is a matching with the most matched vertices. If every vertex has been matched, the matching is *perfect*. With a delocalised compound modelled as a graph, we can think of the Kekulé structure as one, where each atom is adjacent to exactly one double bond. Considering this, the problem is simply to find a *perfect* matching.

The concept of matching is usually applied to bipartite graphs. Here the vertices are divided into two groups and the aim is to find a matching so that every member of one group is matched to a member in another. The compound naphthalene is a bipartite graph and can be used to demonstrate how matching can be used for Kekulisation. Firstly, the vertices are rearranged into two groups (Fig. 5.5). Note that the connectivity is preserved.

There are three distinct perfect matchings, they correspond to three different resonance forms (Fig. 5.6). The cardinality of a matching is increased by finding an alternating path between two unmatched vertices. An alternating path is one where

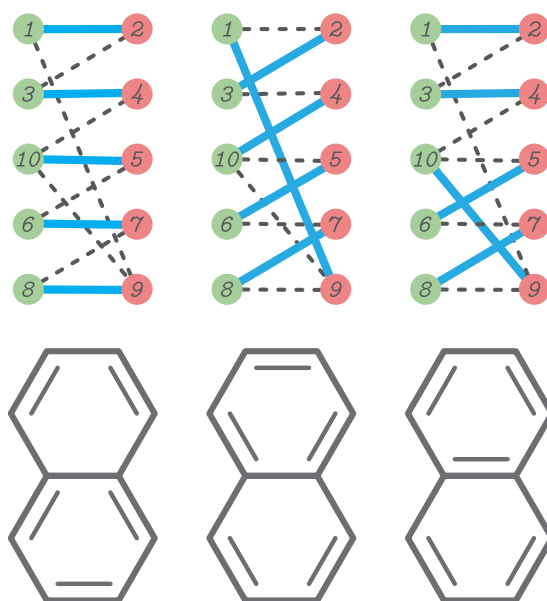


Figure 5.6: The three perfect matchings in naphthalene. Translating the matched edge (blue) to double bond placement provided the three different Kekulé forms.

each edge is alternatively unmatched and matched. For an non-maximum matching (Fig. 5.7a), there may be more than one alternating path (Fig. 5.7b). Once the alternating path is found, the matching is increased by augmenting along the path and inverting the status, matched or unmatched, of each edge. As finding all paths is computationally hard, graph matching algorithms find alternating paths of shortest length. In bipartite graphs newly discovered alternating paths are disjoint and can be found simultaneously. Using this approach, the Hopcroft-Karp algorithm^[224] finds a maximum matching in bipartite graphs in $O(|E|\sqrt{|V|})$.

Not all chemical structures are bipartite a more general algorithm is needed. In non-bipartite graphs, the presence of odd cycles means the alternating path may not be the shortest alternating path in the graph (Fig. 5.8a). However, the alternating paths can still be discovered without the need to explore all paths. Edmonds' blossom algorithm^[225] introduces the idea of contracting odd member cycles (blossoms) to a single vertex. The algorithm then proceeds to find alternating paths in the contracted representation (Fig. 5.8b). Upon identifying the path in the contracted graph, the alternating path is lifted back to the original graph where it can be augmented and the matching improved (Fig. 5.8c). The paths discovered by Edmonds' algorithm are not disjoint and so only one augmenting path can found at a time. This leads to a higher complexity than the bipartite case with $O(|E|^3)$. The use of a more complicated algorithm

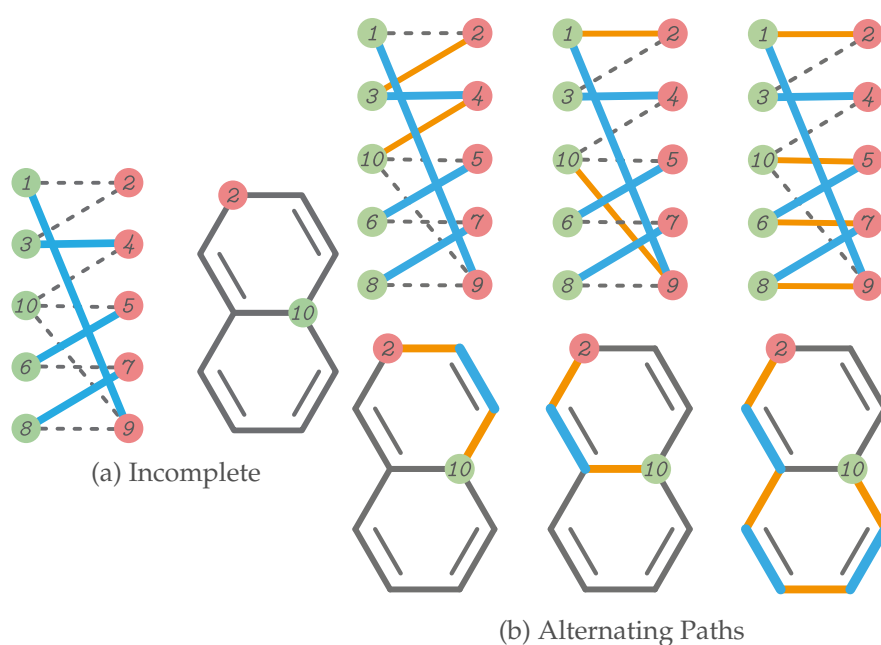


Figure 5.7: **5.7a** The matching (blue) is non-perfect and non-maximum. The atoms 2 and 10 are unmatched and non-adjacent. **5.7b** The matching can be maximised by augmenting one of the alternating paths. Once augmented the unmatched single bonds (orange) become matched whilst the matched double bonds (blue) become unmatched. Matched edges not in the augmenting path remain unchanged. Augmenting along each path produces a different perfect matching.

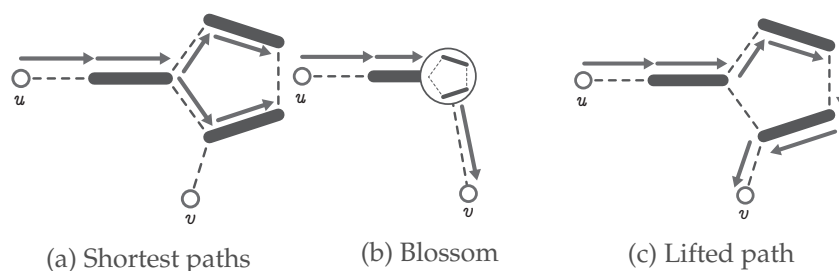


Figure 5.8: A directed shortest path from u to v can not be found directly (5.8a). By contracting the odd-member cycle (a blossom) and treating it as a single vertex a shortest path can now be found (5.8b). The path is lifted back up to the original graph for augmentation.

reduces the complexity to the same as the bipartite case^[226,227].

Restricted matching problem

The restricted matching problem was introduced by Hansen P and Zheng M^[228] to find a matching in a subset of vertices for Kekulisation. In Kekulisation this reflects that only part of a compound may be delocalised and that atoms that contributed a lone pair are not adjacent to a double bond. After using Edmonds' blossom algorithm to identify a maximum matching in the complete graph, the subset of vertices are checked. If the matching in the subset is not perfect, a virtual vertex is introduced and edges are removed from the complete graph. Edmonds' blossom algorithm is then re-run and the process repeated until no more alternating paths are discovered. Working on the whole graph, the algorithm may also match edges of vertices not considered in the subset.

Alternating paths in chemical graphs

In addition to Kekulisation, the theory of graph matching can be of use in the efficient identification of single proton shifts. Migration of a proton can be restricted to 1,3 and 1,5 shifts but longer shifts are feasible and used^[220]. Similar to the Kekulisation, we can identify a candidate shift by finding an alternating path from an atom that can donate an hydrogen to one that can accept it. The alternating path may not be the shortest path. By contracting odd cycles, longer alternating paths can be found. By attaching a virtual vertex to the acceptor, it is possible to efficiently find alternating paths of any length and identify single proton migrations without exploring a poten-

tially exponential number of paths. Additionally, InChI generation uses alternating paths in the “hard” proton removal and additional steps of normalisation^[163].

5.2.3 Fast Kekulisation

Although bounded by a polynomial, Edmonds’ blossom algorithm has a relatively high complexity, $O(|E|^3)$. It is therefore not desirable to perform multiple invocations as is shown in the restricted matching problem. The restricted matching problem also matches edges that do not have endpoints in the subset. These can be removed after processing but demonstrates that redundant alternating paths are being discovered.

A small modification allows Edmonds’ blossom algorithm to identify a matching in a subset of vertices directly. When the alternating paths are being discovered, only edges in which both endpoints are in the subset are considered. We can therefore directly find a perfect matching in the subgraph that covers the vertices in our subset. Since the algorithm does not require a connected graph, the subgraph can be specified without modifying the original input. The subset of vertices is specified as a mask of which atoms to include in the matching problem. No modifications, such as edge removals or vertex additions are needed. This results in a very fast implementation.

Identifying unmatched atoms

When matching is used to Kekulise a structure, we wish to only place a double bond next to each atom that requires one. These atoms form the subset of vertices in the graph that will be matched. The choice of these atoms is dependent on the aromaticity model used but following simple rules has shown good results. The atoms that do *not* need to be matched are identified by checking the following conditions in sequence:

- 1 Any atom not marked as aromatic.
- 2 An atom already adjacent to a double or triple bond it is not considered unless the atom is either a neutral nitrogen with 3 neighbours or a sulphur with 4 neighbours. The exceptions are required for compounds such as [CHEBI:29136](#) and [ChEMBL1188068](#).
- 3 A carbon cation or anion with three neighbours.
- 4 A nitrogen group (N, P, As, Sb) atom with 3 neighbours or with 2 neighbours and a negative charge.
- 5 A oxygen group (O, S, Se, Te) atom with 2 neighbours.

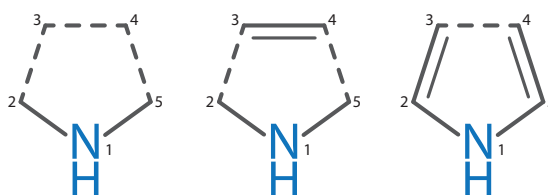


Figure 5.9: Randomly choosing an edge to match in 1H-pyrrole. The matched edge is: 3=4, 2=3, or 4=5. Matching an edge between 2=3 or 4=5 means 3=4 is no longer available.

These rules are by no means complete but allow Kekulisation on the majority of organic compounds in several tested data sets. Problematic compounds encountered in these datasets will be presented later. The identification of the atoms can be efficiently implemented in $O(|E|)$ time by scanning over the bonds and using a switch on each element type. The encoding does not consider triple ($\pi\pi$) bonds that have been delocalised but will correctly assign bonds when the triple bond has been explicitly indicated ([ChEMBL423544](#)). The higher atomic number elements are included for corner cases such as tellurophene ([ChEBI:30858](#)).

Greedy matching

Edmonds' algorithms only seeks to maximise a matching, and so the initial matching need not be empty. Using a greedy procedure, an initial matching can be constructed that reduces the number of iterations needed by Edmonds' algorithm. The greedy matching may even find a perfect matching. With the restricted matching problem a randomly generated matching is suggested^[228]. Unfortunately this means that even for some simple cases the matching may need maximising when a more targeted approach could avoid invoking Edmonds' algorithm.

When performing a random greedy matching on pyrrole, the first matched edge can be either 3=4, 2=3 or 4=5 (Fig. 5.9). If the edge 3=4 is chosen, the matching will need maximising. However, if 2=3 or 4=5 is chosen first, the next matched edge cannot be 2=3 and so the matching will be perfect. In this example, assuming a random order on the vertices, the more computational demanding Edmonds' algorithm is only invoked one in three times. An improved greedy procedure can guarantee a perfect matching of pyrrole.

After determining the set of unmatched vertices, the degree of adjacent vertices (in the unmatched subset) is determined. All vertices with degree one are then selected and

matched. The degree of all vertices adjacent to the other matched vertices is decremented. Once no more unmatched vertices have degree one, an edge is randomly selected and matched, and the adjacent vertices, degree decremented. This may mean there are again vertices with degree one that can be matched. Edges are matched until there are not available edges. If all edges are matched, Edmonds' algorithm is not invoked. Vertices with degree one can be maintained in a stack.

Storage of matchings

Although the matching is translated into double bond placement on a compound, the match can be stored as a single array of integral values. This is possible because each vertex can only be matched with exactly one other. A matching of the edge between vertex u and v can be stored by setting the value at index u to v and vice versa. As it may be expensive to update the bond labels in our graph structure, storing matches separately is preferred.

5.2.4 Unique tautomer generation

The Sayle and Delany algorithm^[190] provides a global approach to tautomer generation. A key part of the algorithm is Kekulisation. Initially, the algorithm assigns atom types that identify a role (i.e. donor or acceptor) for each atom. The atom typing can be modified to enable or disable donation of hydrogens from specific elements. The inclusion of carbon donors and acceptors (i.e. keto-enol) increases the number of generated tautomers and is disabled by default.

Having identified all the donors in a conjugated systems, the protons are then removed. Each hydrogen is then reassigned using a backtracking procedure by trying each atom first as a donor and then as an acceptor (Apdx. D.2). When all atom roles have been assigned and if a Kekulé structure can be found, a tautomer has been generated. To find another tautomer, the search backtracks and attempts other combinations of donors and acceptors.

Imposing a canonical ordering on how donors and acceptors are visited ensures that the first tautomer generated is always the same. The algorithm can therefore produce a unique representation without generating every tautomer. The Sayle and Delany algorithm was implemented using the fast Kekulisation based on graph matching.

5.3 Results and discussion

Table 5.2: Datasets used in algorithm evaluation.

Database	Size	Available
ChEBI 108	26,790	www.ebi.ac.uk/chebi
Zinc (frag)	504,074	zinc.docking.org/
ChEMBL 17	1,318,180	www.ebi.ac.uk/chembl
Zinc (leads)	5,135,179	zinc.docking.org/
Zinc (drug)	11,985,234	zinc.docking.org/

The algorithms were evaluated on five chemical datasets (Tab. 5.2). These datasets provide a download of structures as aromatic SMILES. All runtimes were measured on an Intel Core i7 CPU (2.66 GHz) with 8 GB of RAM.

5.3.1 Aromaticity

The CDKHuekelAromaticityDetector (1.4) was compared to the new implementation (1.5) with different electron donation models (Table. 5.3). Measured times exclude input and atom typing. Atom typing is only required for methods based on CDK atom types. The 1.4 version here includes faster detection of all rings; it was previously a lot slower than presented. Despite this, a significant performance improvement is still seen. The time taken to process Zinc (drug) with 1.5 is quicker than 1.4 took to process ChEMBL, a tenth of the size. The errors n_{err} reported for the 1.5 method are due to undefined values for orbital hybridisation. This occurs when an atom type is not found and an exception is raised; these were previously ignored in 1.4.

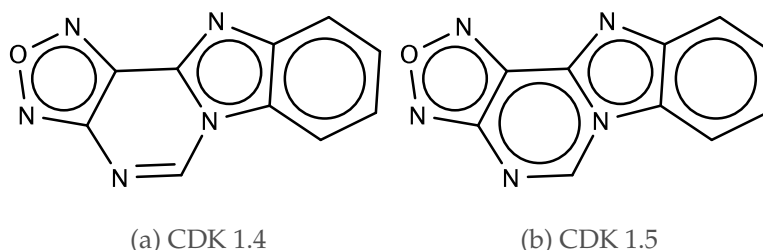


Figure 5.10: Aromaticity perception of [ZINC00032826](#) in previous (1.4) and latest (1.5) CDK versions. The CDK 1.4 only checked larger envelope rings when the minimum cycle basis had less than four members.

Table 5.3: Comparison of old and new aromaticity perception. For CDK 1.5, the suffix of CDK, Daylight, and Pi is referring to the electron donation model.

Dataset	Method	n_{err}	n_{arom}	n_{arom_atom}	t (total)	t (ms / structure)
ChEBI 108	CDK 1.4	0	11011	115505	6 s	0.214
	CDK 1.5 CDK	489	10838	112026	1 s	0.042
	CDK 1.5 Daylight	0	11758	126510	≤ 1 s	0.031
	CDK 1.5 Pi	0	10082	100196	≤ 1 s	0.019
Zinc (frag)	CDK 1.4	0	401168	2982607	42 s	0.084
	CDK 1.5 CDK	3	401166	2983393	7 s	0.013
	CDK 1.5 Daylight	0	411466	3126364	5 s	0.009
	CDK 1.5 pi	0	310616	2065876	3 s	0.006
ChEMBL 17	CDK 1.4	0	1225002	16660848	5 min 40 s	0.258
	CDK 1.5 CDK	2029	1223193	16640844	37 s	0.028
	CDK 1.5 Daylight	0	1235439	17420718	29 s	0.021
	CDK 1.5 Pi	0	1184611	13918215	22 s	0.016
Zinc (lead)	CDK 1.4	0	4838811	46899361	12 min 30 s	0.142
	CDK 1.5 CDK	0	4838811	46909231	1 min 43 s	0.020
	CDK 1.5 Daylight	0	4879994	48715121	1 min 15 s	0.014
	CDK 1.5 Pi	0	4102196	31745422	55	0.010
Zinc (drug)	CDK 1.4	0	11473735	127715175	30 min 51 s	0.154
	CDK 1.5 CDK	5	11473733	127760944	4 min 4 s	0.020
	CDK 1.5 Daylight	0	11538044	133285822	3 min 4 s	0.015
	CDK 1.5 Pi	0	10331816	93116350	2 min 17 s	0.011

Excluding structures with unrecognised atom types, the number of structures detected as aromatic (n_{arom}) is similar using the model based on atom types in 1.4 and 1.5. However, a difference is seen in the number of atoms identified as aromatic (n_{arom_atom}). This is because the newer implementation checks all rings, regardless of the size of the minimum cycle basis (Fig. 5.10). Using a different set of cycles, the old implementation can be exactly replicated.

5.3.2 Kekulisation

Invalid inputs

If a perfect matching cannot be found the input cannot be Kekulised and is flagged as invalid. The primary reason for invalid input is a deficiency in identifying which atoms are adjacent to double bonds. A decision must be made as to whether an atom is adjacent to a double bond. If an atom has abnormal valence the decision can be incorrect.

Of the tested datasets, 14 compounds in ChEMBL and 10 compounds in ChEBI were

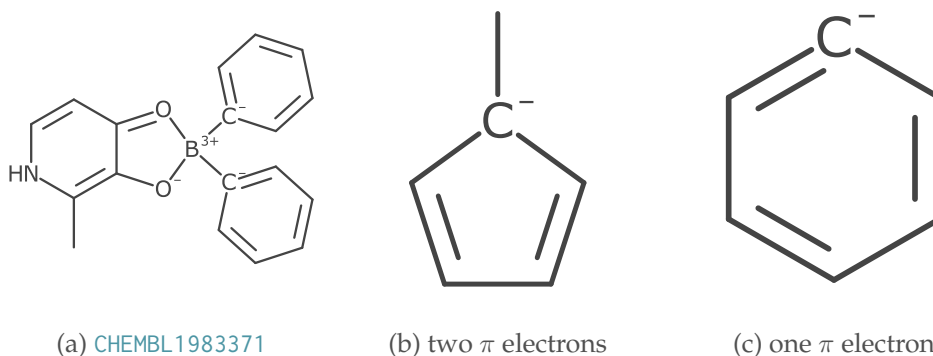


Figure 5.11: Incorrectly delocalised tetraivalent carbon anions cannot be Kekulised. The ChEMBL entry is implausible and has been incorrectly delocalised when the SMILES was generated. The loss of information and the implausibility means the representation stored in ChEMBL (5.11a) can not be retrieved. When adjacent to three bonds the algorithm presumes two π electrons were contributed (5.11b). In fact there was only one. This is only expected when adjacent to two atoms (5.11c).

found that could not be Kekulised. All compounds in the Zinc subsets could have electrons assigned. The compounds from ChEBI and ChEMBL that could not be localised were investigated and verified using the Daylight DEPICT service^[229]. The service generates a depiction for SMILES and indicates if the input is invalid.

In ChEMBL all the failures were found to be due to tetraivalent carbon anions (Fig. 5.11a). Due to their abnormal valence the atoms were not selected as being adjacent to a double bond. The subset identifies a carbon anion as either contributing two π electrons from its lone-pair (Fig. 5.11b) or one π electron (Fig. 5.11c). The correct representation (Fig. 5.11a) can only be obtained by including both carbons in the subset. Given that for normal valence this would not be the case, multiple Kekulisations would need to be tested with different combinations of atoms in the subsets. Testing all possible combinations of atoms in the subset is feasible for a small number of atoms (2^n) but could produce multiple valid solutions that would then need to be selected. All 14 examples from ChEMBL were marked as invalid by the Daylight DEPICT service.

The 10 invalid inputs from ChEBI were due to: dipolar bonds (6), radicals (3) and pseudo atoms (1). The dipolar bonds were found in iridium (CHEBI:52749, CHEBI:52748) and europium (CHEBI:52729) coordinated complexes as well as pseudocoenzyme B12 (CHEBI:48572, CHEBI:48573, CHEBI:48568).

The problem is similar to the example in ChEBML. In each case a neutral tetraivalent nitrogen has been delocalised. The algorithm identifies a nitrogen with 3 connections

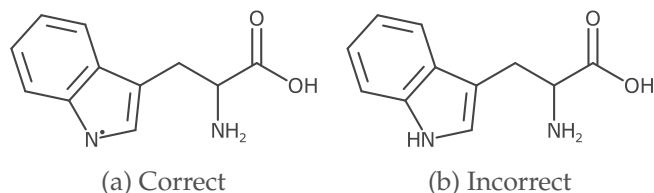


Figure 5.12: Interpretation of tryptophanyl (CHEBI:32730). The SMILES input for this structure is, CC(C)c1cccc2c1[n]cn2n1cccc1. The ChEBI entry has a radical and was incorrectly delocalised in the SMILES string.

which indicates the nitrogen is not adjacent to any double bonds. These problems are the result of a loss of information between formats where dipolar bonds have been converted to sigma bonds. Three tryptophanyl compounds (CHEBI:32730, CHEBI:32712 and CHEBI:32730) could not be Kekulised due to the presence of a radical. As radicals cannot be explicitly represented in SMILES an aromatic nitrogen with two neighbours and no implicit hydrogen is seen as being adjacent to a double bond. This tryptophanyl example demonstrates why it is problematic to try and correct for invalid input. The possible fix to this compound from ChEBI could be to add a hydrogen to nitrogen. As the original input structure actually contained a radical, this would not provide the correct structure (Fig. 5.12).

The graphite (CHEBI:33418) entry could not be interpreted due to the use of pseudo (unknown element) atoms. As noted by the OpenSMILES^[230] specification, an '*' (unknown) atom can be part of an aromatic ring if there is an element that could be used in place of it to make the ring aromatic. When localising bonds, we must then consider the '*' as variable and allowed it to both matched and unmatched. This is currently beyond the scope of the algorithm. With the exception of graphite, the other nine compounds were also not accepted as valid by the Daylight DEPICT service.

Performance

The performance of the Kekulisation was evaluated on the five datasets. The time for each dataset was broken down to compare how much of an impact the use of the greedy matching procedure had. The greedy algorithm was able to find a perfect matching covering the unmatched subset for more than 80% of delocalised compounds (Table. 5.4). In these cases, the more expensive maximum matching is not invoked. The variation between data sets is likely to be due to the use of different aromaticity models in their creation. The absolute time taken to find a maximum

Table 5.4: The total number of compounds, n , those that some delocalised part, n_{arom} , and those that could be Kekulised with only the greedy algorithm, n_{greedy} .

Dataset	n	n_{arom}	n_{greedy}	$\frac{n_{greedy}}{n_{arom}}$
ChEBI 108	26,790	11,705	10,948	0.93
Zinc (frag)	504,074	411,403	356,016	0.86
ChEMBL 17	1,318,180	1,224,895	1,140,669	0.93
Zinc (leads)	5,135,179	4,879,898	4,062,099	0.83
Zinc (drug)	11,985,234	11,537,839	9,787,357	0.81

Table 5.5: Breakdown of absolute time taken (seconds) to Kekulise the compounds.

Dataset	Greedy	t_{conv}	t_{type}	t_{greedy}	t_{edm}	t_{total}
ChEBI	off	0.15	0.05	-	0.14	0.34
	on	0.14	0.05	0.04	0.02	0.25
Zinc (frag)	off	1.66	0.52	-	1.10	3.31
	on	1.57	0.63	0.34	0.15	2.69
ChEMBL	off	9.03	3.35	-	6.13	18.57
	on	7.47	2.86	1.19	0.25	11.76
Zinc (leads)	off	22.35	6.85	-	13.48	42.91
	on	20.32	6.15	3.77	1.39	31.63
Zinc (drug)	off	59.79	25.89	-	39.31	125.57
	on	56.64	18.77	10.96	3.35	89.71

matching using Edmonds’ algorithm, t_{edm} , was reduced with the use greedy algorithm, t_{greedy} (Table. 5.5). The total time spent matching ($t_{greedy}+t_{edm}$) was reduced from 6.13 to 1.44 seconds for ChEMBL and from 39.31 to 14.31 for Zinc (drug). The time taken to identify unmatched atoms, t_{type} , takes longer than the matching. The time to convert, t_{conv} , from the CDK native AtomContainer to an adjacency list is also seen to be the bottleneck. Even including the conversion, a time of 11.76 seconds is a minimal overhead when reading 1,318,180 compounds from ChEMBL.

Performance comparisons to existing algorithms are difficult due to other improvements. In particular the other procedures rely on ring perception and atom typing which have both been optimised. Despite this, when testing on ChEMBL 17 the FBOT and DBST both took 281 seconds, white the ATASC took 77 minutes.

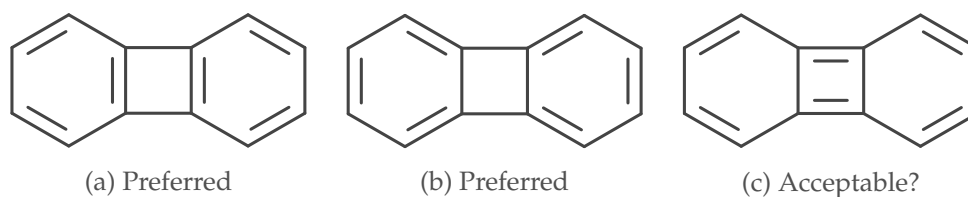


Figure 5.13: Different Kekulé representations of biphenylene. Preserving the familiar benzene substructure is preferred.

Limitations

One limitation in the approach is demonstrated in, biphenylene, an example presented by Roger Sayle^[223]. The representations where all double-bonds are present in the six member rings (Fig. 5.13a, 5.13b) are preferred due the recognisable benzene. The Kekulisation matches all at once can result in the less desirable representation (Fig. 5.13c) being found. The assignment that is found depends on the ordering of the atoms and bonds. If the single bonds between the benzenes are provided in the input the matching algorithm could be modified to localise the bonds in each connected aromatic system separately obtaining the preferred representation.

Canonical Kekulé representation

Delocalised aromatic bonds cannot always encode the multiple equivalent representations of a compound. As 1-methylpentalene does not fit any aromatic model its bonds are not delocalised. With the same atom ordering, two distinct SMILES (Fig. 5.14) can be generated, with the compounds having the same InChI.

When generating Kekulé SMILES it has been suggested by Weininger *et al*^[102], the OpenSMILES specification^[230], and Universal SMILES^[231] that double bonds are visited first in the traversal. Starting at any atom in 1-methylpentalene and following this rule still leads to different traversals and different SMILES for the same ordering of atoms. This is even true when a compound contains a benzene ring^[232]. Instead of using aromaticity perception to delocalise bonds, a unique Kekulé structure can be assigned. Given different double bond locations the algorithm assigns a unique Kekulé form.

The matching obtained by the Kekulisation algorithm is dependent on the ordering of the atoms. A unique representation can then be obtained by uniquely ordering the

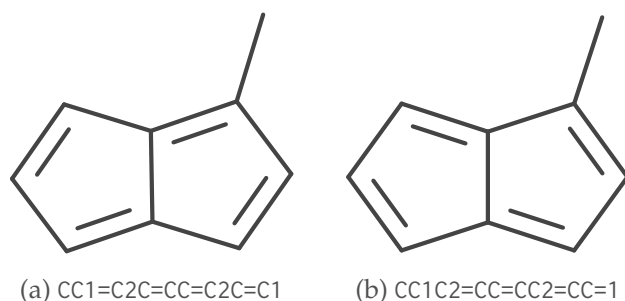


Figure 5.14: Different Kekulé representations of 1-methylpentalene (CID 21019565). The number of electrons in any ring does not equal $4n + 2$ and is not delocalised.

atoms and re-Kekulising a compound. The selection of the atoms to be matched is simply any atom that is adjacent to exactly one double bond. Double bonds that have a stereochemical configuration can optionally be included.

5.3.3 Unique tautomer generation

The generation of unique tautomers was evaluated on compounds from the ChEBI 108 and ChEMBL 17 databases.

ChEBI roundtrip

To ensure the same unique tautomer was generated, up to the first 20 tautomers were enumerated and randomised for every compound in ChEBI. Hydrogens attached to carbons were disabled (i.e. no keto-enol). 9,553 ChEBI entries were found to have more than one tautomeric form and when enumerated this produced 48,872 distinct entries. A canonical tautomer was then generated for all 48,872 entries and all 9,553 unique entries were successfully retrieved.

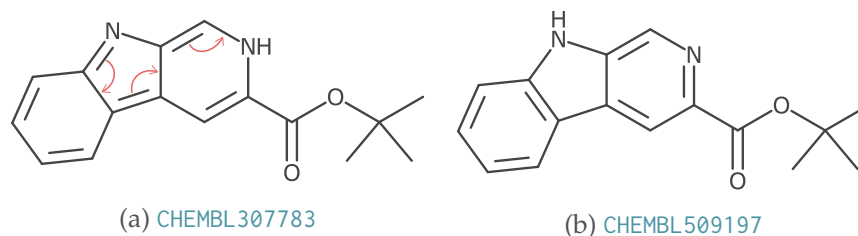


Figure 5.15: A 1,7-shift of a hydrogen. The structures are encoded as different InChIs. CHEMBL509197 has been removed in ChEMBL 18.

Identifying tautomers in ChEMBL

The validation on ChEBI showed tautomers could be successfully generated and rectified. Although ChEBI does contain tautomers (Chap. 2) and they are annotated in the ontology, many were zwitterionic and beyond the scope of the algorithm. The larger ChEMBL (release 17) database was inspected for tautomers using the Sayle and Delany implementation. Discovered tautomers were compared to the InChI for validation.

Entries were then grouped separately by both the unique tautomer and main layers of the InChI. Groups that were found to only differ in stereochemistry or isotopes were removed. To mimic the default configuration of the Standard InChI, mobile hydrogens attached to carbons were again ignored. The groups were then inspected for entries that each algorithm did not identify as potential tautomers.

Assigning a unique tautomer identified 1,477 sets of tautomers, 586 of which were found by the standard InChI and a further 686 by enabling 1,5 shifts (15T option). A selection of the remaining 207 sets were manually inspected and found to be tautomers with longer hydrogen shifts (Fig. 5.15). The standard InChI identified 704 sets of tautomers; initially only 585 were also found by the canonical tautomer. 87 missed tautomer sets were seen to be false positives due to normalisation of the charge separated sulfoxide groups¹. Accounting for this normalisation, the InChI then identified 617 sets of tautomers of which 586 were identified by canonical tautomer. The remaining 31 sets were found to be due to other charge normalisations not applied by the canonical tautomer (Fig. 5.16). Although these do involve the relocation of a hydrogen, the normalisation is not performed during the tautomer detection process of the InChI^[163].

¹The InChI standardises $[S+](O-)$ to $S(=O)$.

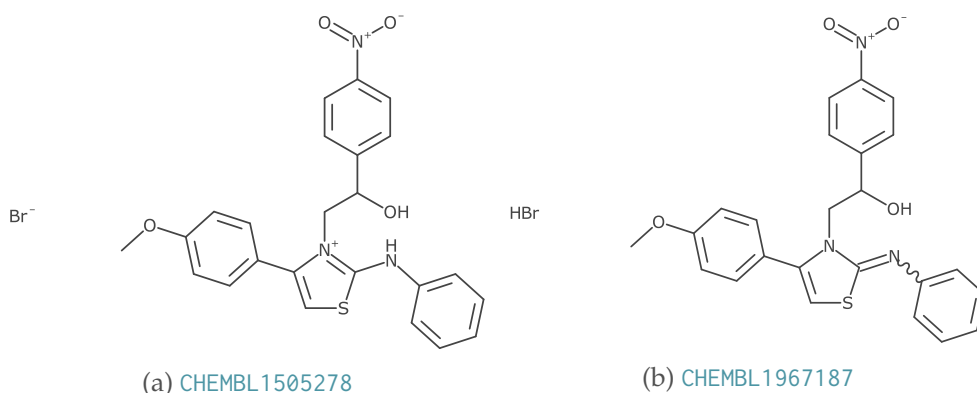


Figure 5.16: A normalisation performed by the InChI allows these two structures to be identified as equivalent. The normalisation is performed through neutralisation.

Table 5.6: Time taken, t , to generate a unique tautomer for all compounds in a given dataset. Time taken for canonical generation in the valence state combination (VSC) model was reported by Urbaczek *et al*^[233].

Dataset	t (s)	t (ms/cmpd)	t_{vsc} (ms/cmpd)
ChEBI 108	1.48	0.055	-
Zinc (frag)	4.70	0.009	-
ChEMBL 17	30.93	0.023	0.73
Zinc (leads)	75.67	0.014	0.31
Zinc (drug)	201.23	0.016	-

Performance

The efficiency of the underlying Kekulisation algorithm allows rapid generation tautomers, even for the larger datasets (Table. 5.6). Comparison to the existing algorithms based on the CDK could not be made as both algorithms stalled for even small inputs.

A method was recently published that used the Sayle and Delany algorithm to enumerate and canonicalise tautomers, resonance forms, and protonation states^[233]. Run-times for canonical generation were reported for two of the evaluated datasets. Accounting for more than just tautomerism is more complicated and could account for the slower runtime. The reported times were measured on an Intel Core i5-3570 CPU (3.40 GHz).

5.4 Conclusions

Efficient and accurate algorithms have been implemented and included in the Chemistry Development Kit. The algorithms can be used in the aromaticity perception, Kekulisation, canonical Kekulé representation, and tautomer generation.

A fast and configurable aromaticity algorithm has been created. The improved performance allows much larger datasets to be standardised in a fraction of time. The configuration of parameters allow the method to replicate different aromaticity definitions. Definitions that allow the summing of electron counts from smaller rings and variable contribution cannot be replicated (Apdx. D.1) but it is not clear how useful these features would be.

The fast Kekulisation is now applied automatically to all SMILES input and allows correct interpretation by other methods in the library.

A limitation in the Kekulisation was seen for a small set of compounds. The majority of problems were due to atoms with abnormal valence being delocalised. These could be corrected by trying to match with all possible atom combinations. There is however no guarantee the correct output is obtained. As the number of problematic compounds is small and usually the result of a bug or corner case in aromaticity detection, no attempt is currently made to correct the input. The radical example in ChEBI was produced with the JChem. Until version 6.2 (Q1 2014) the library would still produce the output but will also reject it as invalid. It should also be noted that older versions of the CDK were in part responsible for producing erroneous output.

The default output of the CDK unique and absolute SMILES generation is now a canonical Kekulé encoding. This overcomes the limitation of aromaticity to represent equivalent compounds. Aromatic SMILES can be problematic and an output may be generated that cannot be localised. Output as a Kekulé SMILES avoids these problems with re-interpretation and still allows the output to be used as a unique key.

The implementation of a fast tautomer generation algorithm has proved useful in the identification of tautomers and was utilised in Chapter 2. There are several improvements that could be made to the implementation. The assignment of donor and acceptor roles could be improved so as to avoid modifying bonding around atoms with a stereo configuration. Alternatively, as with the InChI, the tautomer generation could indicate stereocentres that may undergo transition, and should therefore not have a

stereo configuration assigned. Limiting of the hydrogen shift distance is also possible. However, this undermines the utility of a global approach.

The ordering of the atoms attempts to generate low energy tautomers first but there is no explicit scoring that indicates the most favourable representation. One approach could be to score by aromaticity and the Gibbs energy but such “quick fixes” are unlikely to be useful^[218].

Due to symmetry in a compound, the algorithm may produce two tautomers that are isomorphic (e.g. carboxylate). Filtering out these compounds after generation is possible but requires storing and checking previous results. It is possible to identify these tautomers by inspecting the symmetry classes of the donor and acceptor atoms.

5.5 Availability

The aromaticity and Kekulisation algorithms are already included in the CDK code-base, <http://cdk.github.io/cdk>. The Java class names are:

- `org.openscience.cdk.aromaticity.Aromaticity` – front-end API for assigning aromaticity
- `org.openscience.cdk.aromaticity.ElectronDonation` – access to electron donation models
- `org.openscience.cdk.aromaticity.Kekulization` – front-end API for Kekulisation
- `org.openscience.cdk.graph.Matching` – storage of graph matching, includes the greedy algorithm
- `org.openscience.cdk.graph.EdmondsMaximumMatching` – Edmonds’ blossom algorithm to maximise a matching over a subset of vertices
- `org.openscience.cdk.tautomer.*` – package containing role assignment and backtracking generation

Chapter 6

Stereochemistry

6.1 Introduction

6.1.1 Motivation

Representing compounds as only connections in a graph fails to capture the spatial arrangement of atoms. Two compounds may have the same formula and connectivity but a different spatial arrangement. When two compounds have spatial arrangements that cannot be superimposed, they are referred to as stereoisomers. Without accounting for the spatial arrangement, metabolites like glucose, galactose, and mannose, are seen as identical. In Chapter 2 several different metabolites retrieved the same candidates when stereochemistry was missing. As many enzymes are specific and sensitive to a particular stereoisomer, representing and handling stereochemistry is essential for studying metabolism.

6.1.2 Stereocentres

The stereochemistry of a compound is closely coupled to molecular geometries, rotational symmetries of these geometries and the ordering of atoms. A common molecular geometry found in organic chemistry is tetrahedral. This geometry is found in neutral carbons with four neighbours. A tetrahedron has 12 rotational symmetries but 24 (4!) different ways that the neighbours can be ordered. There are then two

(24/12) distinct arrangements of atoms that are possible. Each arrangement is a reflection symmetry and is not superimposable.

A tetrahedral atom is a stereocentre if there are four distinct neighbours. If two or more neighbours are equivalent (experience the same environment), some of the 24 different orderings of atoms will be identical. When the number of the unique orderings is then less than or equal to the 12 (number of rotational symmetries) the atom is not a stereocentre. The neighbours may be atoms or lone pairs and the equivalence may be conditional of topographical.

As well as other atom-centric stereocentres, a bond may have distinct spatial configurations. Since a double (π) bond cannot rotate there are distinct orientations of the connected atoms. The position of a substituent (connected to one atom in the double bond) relative to a substituent of the other atom can be either on the same or opposite side of the bond. Similar to the tetrahedral stereocentre, when two substituents connected to the same atom are equivalent, it is not a centre of stereochemistry.

The configuration around a double bond may be referred to as: cis/trans, E/Z, or geometric isomerism. The first two are really naming schemes and so the term geometric will be used in this chapter.

Other forms of stereochemistry are possible but the tetrahedral and geometric stereocentres are by far the most common in organic chemistry.

6.1.3 Describing stereochemistry

The configuration of stereocentres can be described and represented in multiple ways. If the three-dimensional coordinates are present, the exact spatial relation of atoms is known.

In two-dimensional depictions, the planar orientation of geometric isomers can be represented directly with different spatial positions. Tetrahedral centres have a 3D geometry and require the use of nonplanar bonds to indicate atoms above and below the plane of the depiction. Tetrahedral centres may also be represented by one or more perspective projections.

Linear notations such as SMILES, InChI, and nomenclature name the spatial configuration relative to some ordering of the atoms. The ordering may be a local reference, such as the atom numbers, or a global reference like a canonical ordering or the Cahn-

Ingol-Prelog (CIP) rules. The CIP system provides a set of rules to rank substitutes adjacent to stereocentres. By ranking the atoms a label such as *R* or *S* (for tetrahedral centres) can be assigned. This is the primary method of indicating stereochemistry in nomenclature.

6.1.4 Objectives

Accurate handling of stereochemistry in the CDK was not an initial design goal. There is support for different aspects but these are generally disconnected and do not cooperate.

Tetrahedral configurations are handled by an `AtomParity` data structure that stores a central atom, neighbours and the winding configuration. The `AtomParity` is read and written by the InChI and CML formats.

Similarly, the `TetrahedralChirality` data structure duplicates the data structure seen in `AtomParity` and is primarily used to determine a Cahn-Ingold-Prelog (CIP) label. A stereo parity number can be stored as an attribute on the atom and read from a CTfile but is not used or modified.

Generation of SMILES output would examine the 2D coordinates to determine the configuration of tetrahedral carbons and geometric configuration can be manually specified. The SMILES generation incorrectly encoded the tetrahedral configurations in rings and attempts to encode trigonal bipyramidal and octahedral stereocentres (difficult). When reading SMILES, `TetrahedralChirality` was created.

A new `DoubleBondStereochemistry` data structure was recently (end 2012) added to accompany the `TetrahedralChirality` data structure for interpreting InChI identifiers.

In many cases it is not possible to preserve the stereochemistry when reading and writing between formats. Even in the same format (SMILES) the input creates data structures but the output needed a 2D depiction. Generated 2D depictions are laid out without any indication of tetrahedral stereochemistry and geometric isomers are randomly depicted (e.g. always trans).

To adequately handle stereochemistry, it must be possible to convert and reason about the different representations. During this project the handling of stereochemistry in the CDK has been overhauled, unified, and enhanced. Five areas are addressed in

this chapter: representation, identification (from 2D and 3D coordinates), depiction, comparison, and description.

The techniques for handling stereochemistry are relative well established. This chapter collects and describes methods for above tasks and highlights required considerations. The identification, depiction, and description are evaluated by comparison to other toolkits.

6.2 Methods

All algorithms are free and openly available, links to the source code are listed at the end of the chapter.

6.2.1 Representation

To improve the handling for stereochemistry, the representations first needed to be standardised. The AtomParity was removed and existing usages were replaced with the newer TetrahedralChirality. An ExtendedTetrahedral structure was also added for representing the arrangement around an even number of cumulated double bonds.

The existing DoubleBondStereochemistry was adequate for representing geometric isomers of a single double bond. An odd number of cumulated bonds is again planar and a geometric isomer but this is not yet included.

These data structures are referred to in the CDK as StereoElements and store the local arrangement of atoms in a structure.

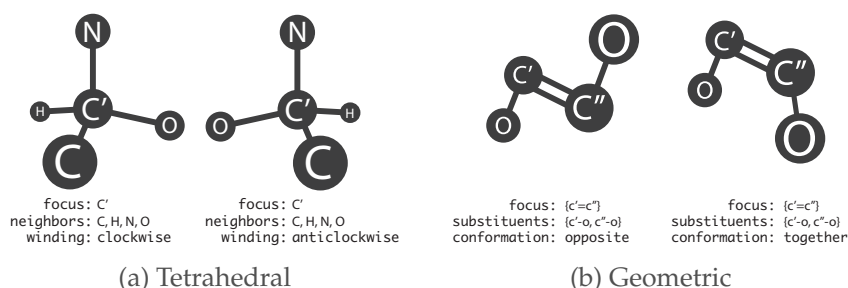


Figure 6.1: Data structure representations of tetrahedral and geometric stereochemistry. The tetrahedral stereochemistry is stored in an atom centric manner. The geometric stereocentre is stored in a bond centric manner. The same tetrahedral configuration may be represented in twelve different orderings, only one is shown.

Tetrahedral stereochemistry Tetrahedral centres are described by a central atom, four neighbours and a winding. Looking from the first neighbour towards the central atom, the other three neighbours proceed either clockwise or anti-clockwise winding (Fig. 6.1a). There are 12 different orderings that can represent the same stereocentre with the same winding. The extended tetrahedral centre is similar except the cumulated atom is the central atom and the four neighbours are connected to either terminal

atom.

Geometric stereochemistry Geometric stereochemistry is represented by storing a double bond, two substituent bonds, and a conformation. Each substituent bond is attached to either atom of the central double bond and the conformation is either opposite or together (Fig. 6.1b). When an atom of the double bond has two substituents, only one is stored. When both atoms have two substituents, there are four different orderings that can represent the same stereocentre.

Implicit atoms The description of the tetrahedral stereochemistry requires four atom references to be stored as the neighbours. In a hydrogen suppressed graph, one of the neighbours may be a hydrogen that does not have an explicit node. Previously all tetrahedral representations required four explicit atoms and attached hydrogens always had to be explicit.

To handle hydrogen-suppressed structures, the central atom is now used as a placeholder in the neighbour list. An alternative would be to use a phantom atom but using the central atom simplifies comparison and depiction operations. The central atom is also used as a placeholder for lone pairs. This allows representation and reasoning about sulfoxide moieties. Two implicit hydrogen atoms may be present in extended tetrahedral centres and so the two terminal atoms are used in place of the single central atom reference. The geometric stereochemistry only requires one substituent and so an explicit node can always be used.

6.2.2 Identification

The stereochemistry representations can be identified and created from a compound with 2D or 3D coordinates. Formats that store 2D or 3D coordinates may optionally mark the stereochemistry. This is often omitted and so the stereochemistry needs to not only be determined but also located.

Determinant method for spatial configuration The configuration of a tetrahedral stereochemistry is computed as the signed volume of the tetrahedron formed by coordinates of the four neighbour atoms^[234]. The signed volume is computed with a

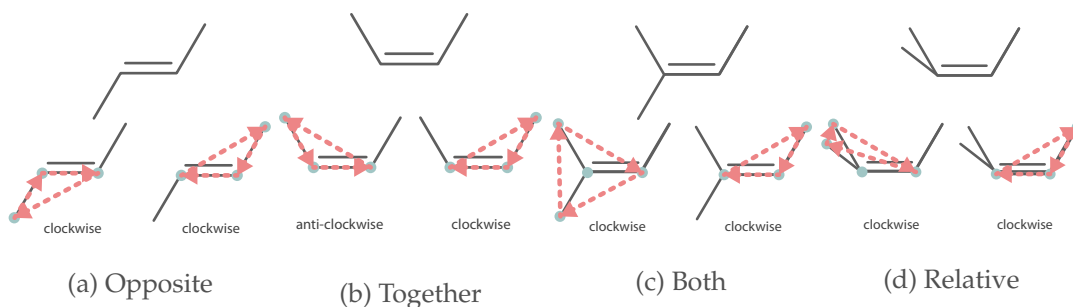


Figure 6.2: The configuration of geometric isomers can be determined by calculating the parity of triangles formed by the substituents and the terminal atoms of the double bond. When the triangles have the same winding (clockwise/anti-clockwise) the substituents are opposite (6.2a). When they have different windings, they are the together (6.2b). When more than one explicit atom is present, both are used to form the triangle (6.2c). This allows the computation of relative positions in the rare case that both substituents are on the same side (6.2d).

four-by-four matrix determinant, where the sign indicates the winding of the neighbours. In the presence of an implicit neighbour, the location of the central atom is used. For 2D depictions, each atom must identify whether it is above or below the plane of the depiction whereas 3D coordinates used the absolute values.

The configuration of a double bond can be found by comparing the windings of two triangles formed of each substituent and the other double bonded atom (Fig. 6.2). A three-by-three matrix determinant computes the winding of the triangle in the plane. When the two triangles have the same winding, the substituents are on opposite sides of the double bond (Fig. 6.2a). When the windings are different, the substituents are on the same side (Fig. 6.2b). When two substituents are present, the central atom is not used (Fig. 6.2c). This is needed as inspecting the absolute position of only one substituent does not always identify the correct configuration.

Although rare, two substituents attached to the same atom may be located on the same side. By using both substituents it is possible to determine the relative positions of each (Fig. 6.2d). Detecting double bond configurations in 3D cannot follow the same approach as even a slight twisting of the double bond can cause the windings of one triangle to be inverted and give the incorrect configuration. Since the ambiguous case in 2D (Fig. 6.2d) does not occur in 3D, the configuration can be determined by checking which side of the bond the substituents reside using a cross-product.

Nonplanar bond conventions In 2D depictions, nonplanar bonds are used to indicate an atom is above or below the plane of the depiction. The relative positions above

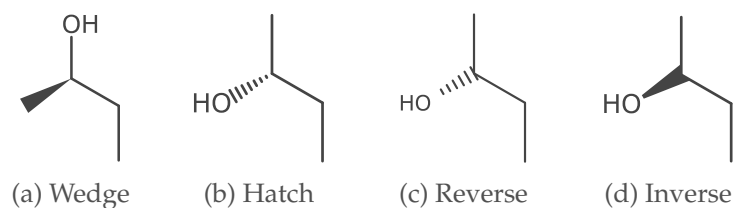


Figure 6.3: Wedge and hatch bond conventions for depicting the stereochemistry in (2R)-butan-2-ol. The most common conventions are 6.3a and 6.3b. The reverse interpretation of hatch bonds mirrors perspective with the narrow end further away and below the plane, 6.3c. Treating the wedge as relative allows the inverse representation where a wedge indicates a down bond relative to the stereocentre, 6.3d.

or below the plane are required when computing the spatial arrangement.

The most common nonplanar bonds are solid “wedge” and broken “hatch” bonds. The de facto convention is that the narrow end of the wedge bond is below the wide end (Fig. 6.3a). For hatch bonds, the narrow end is above the wide end (Fig. 6.3b). The reverse interpretation with the narrow end below the wide end is also used (Fig. 6.3c).

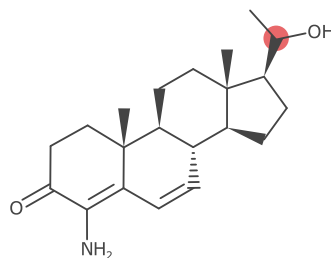


Figure 6.4: Ambiguous stereocentre (CID 67753223), located during development of the hash code (Chap. 2). Does the wedge bond indicate a configuration of the stereocentre at the wide end?

The stereocentre may be located at either end or the wedge or hatch (Fig. 6.3d). This is ambiguous when the bond is connected two stereocentres (Fig. 6.4) and so a convention must be used. PubChem uses the convention that stereocentres are only indicated at the narrow end of the bond^[235].

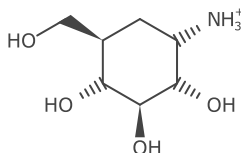


Figure 6.5: The MetaCyc depiction of validamine (CPD-7301) uses the reverse hatch representation.

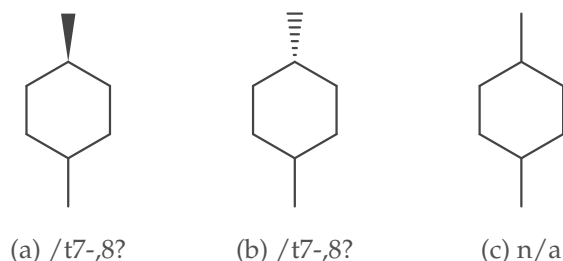


Figure 6.6: Partial stereochemistry encoded by the InChI, 6.6a and 6.6b encode the same identifier. This identifier is different from that encoded by 6.6c. Inverting the bond direction produces the same identifier.

To interpret nonplanar bonds, the IUPAC guidelines for depicting stereochemistry are followed^[236]. These state that the wedge bond points up from the narrow end and hatch bonds point down. Hatch bonds can also be interpreted as pointing down from the wide end (reverse) if the attached atom is unambiguously not a stereocentre (Fig. 6.5). Wedge bonds are not interpreted this way.

Locating stereocentres In order to create the stereochemistry representations from 2D and 3D depictions the atoms and bonds involved must be located. In 2D depictions, the nonplanar bonds can be used as guide for tetrahedral stereochemistry but do not indicate geometric centres. Similarly, with 3D coordinates, no bonds are nonplanar. To locate the stereoelements the environment of each atom and bond is examined and determined as to whether it really is a stereocentre.

To identify stereocentres, the symmetries within a compound must be inspected. Since stereocentres may be inter- and intra- dependent on the configuration of other stereocentres, the symmetry perception is non-trivial. Although it is possible to encode partial stereochemistry of intradependent (Fig. 6.6), it is not useful to encode these in as a normalised representation.

To identify stereocentres, the Cahn-Ingold-Prelog (CIP) rules for uniquely ordering substituents could be used. However, an exhaustive exploration of the compound using the CIP ordering is computationally intensive. As only the location of stereocentres is needed and not an absolute labelling, a modified version of the extended connectivity algorithm can be used^[195]. The method recursively inspect the structure and classifies atoms as either *true* and *para* stereocentres. The *true* stereocentres have neighbours that are all constitutionally different whilst the *para* stereocentres may have constitutionally equivalent neighbours (i.e. Fig. 6.6a). Since all symmetries

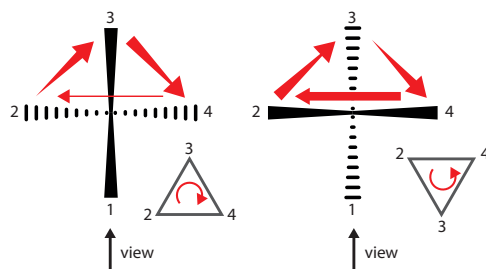


Figure 6.7: Nonplanar labels for a tetrahedral centre with four explicit atoms. Looking from atom 1 towards the focus from a wedge bond, the clockwise triangle formed of red straight arrows translates to a clockwise ordering in 3D. When looking from a hatch bond, the 3D configuration is anti-clockwise.

in the graph are not inspected the *para* stereocentres only indicate potential stereocentres that only have a configuration under certain configurations of other stereocentres. Currently, for its speed, this is the default method used to identify stereocentres but a more comprehensive inspection may be utilised in future.

6.2.3 Depiction

Stereochemistry from linear notations such as SMILES, InChI, and nomenclature is created directly during input. A 2D depiction of the input can then be generated. The depiction is useful not just for visual inspection but is the most portable way to encode stereochemistry in formats such as the CTfile.

Geometric configurations are indicated by the 2D coordinates and so directly dependent on the layout of atoms. Ideally, the geometric configurations would be placed during diagram generation but modification to the existing layout generation was difficult. Instead, geometric configurations are inspected after a diagram is generated and corrected by reflecting one side. A major limitation of this is that geometric stereochemistry present in macrocycles cannot be adjusted. For this reason, identification of geometric isomers in rings is currently disabled.

The indication of tetrahedral and extended tetrahedral centres using nonplanar labels can be applied after the coordinates have been assigned.

To describe the nonplanar label choice, a tetrahedral centre with four explicit atoms will first be considered. The nonplanar labels must always be alternating (Fig. 6.7) but

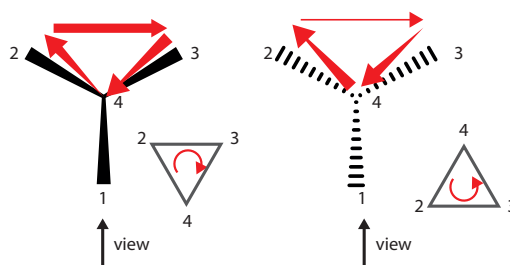


Figure 6.8: Nonplanar labels for a tetrahedral centre with three explicit atoms. As with four atoms, the view from a wedge corresponds to a clockwise 3D configuration (clockwise in the plane). The view from a hatch results in an anti-clockwise 3D configuration (clockwise in the plane).

only one of the bonds needs to be labelled. When tracing the triangle of atoms in a clockwise manner in the plane (red **linear** arrows), the represented configuration (red **circular** arrow) is translated as clockwise when we start at a wedge (Fig. 6.7). Starting at a hatch labelled bond we observed that tracing the position of atoms clockwise in the plane indicates an anti-clockwise configuration in the representation. The correct representation can then be displayed by choosing the start bond direction (wedge or hatch) and placing the other neighbours in a clockwise manner.

To assign labels to atoms with existing positions, the neighbours in the representation are sorted so that they are in clockwise order around the central atom. Recall that the coordinates have already been generated, the first neighbour is arbitrary and it only matters that they proceed clockwise. When sorting, the number of swaps is counted and used to determine the permutation parity. An odd permutation parity (i.e. an odd number of swaps) indicates the stored winding has been inverted, otherwise it remains the same. Starting at the first neighbour, the adjusted winding is used to determine whether to start at a wedge or hatch label. The neighbours are then labelled in clockwise order with alternating wedge or hatch assignment.

Assignment to a tetrahedral centre with three explicit atoms first appears to be simpler. When the neighbours are distributed evenly, the procedure is the same as for the four neighbours but the labels do not alternate (Fig. 6.8). An exception to this is when the placement of the implicit atom is at the corner of the tetrahedron. In structure diagram generation this can occur in rings and the labels must again be alternated. The label that needs to be inverted can be identified by inspecting the triangles formed of clockwise neighbours and the focus. As the coordinates are sorted clockwise, if an

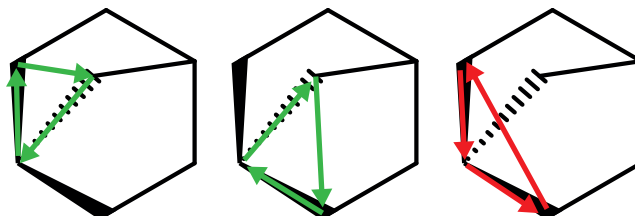


Figure 6.9: Tetrahedral centres in rings may need one nonplanar label inverted, the bond can be identified by finding an anti-clockwise triangle (red) formed by the other two neighbours.

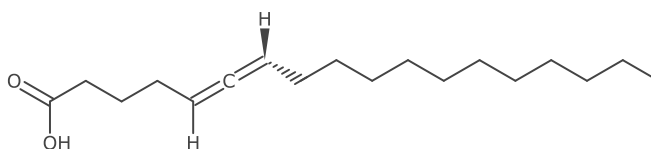


Figure 6.10: A generated depiction of extended tetrahedral stereochemistry in (R)-laballenic acid ([CHEBI:38401](#)).

anti-clockwise triangle is found then the label of the other bond is inverted (Fig. 6.9). The triangle winding is identified with a determinant.

All labels are first determined and assigned to a temporary buffer for selection. One or more bonds must then be selected for the label assignment. Some bonds are more desirable to label than others. The priority attempts to maximise interpretability and minimise ambiguity. The two primary concerns are bonds between two centres and cyclic bonds. As discussed, a nonplanar bond between two centres can be ambiguous as to whether both atoms have a specified configuration. Cyclic bonds are generally avoided as it can be more difficult to manually interpret the configuration. The priority of bonds is: bonds to non-stereogenic atoms, acyclic bonds, bonds to atoms with fewer neighbours, and lower atomic number. The bonds are ordered by this priority and the first unlabeled bond is selected to have a nonplanar label.

Determining the nonplanar labels to indicate extended tetrahedral is identical to the assignment of tetrahedral centres. Although only one bond needs a nonplanar label, other toolkits (such as the InChI) only identify the stereochemistry if two labels are present. Therefore during assignment, two bonds are selected to have a nonplanar label; the bonds must be attached to the same atom (Fig. 6.10).

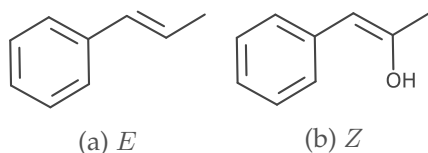


Figure 6.11: Absolute (Cahn-Ingold-Prelog) descriptors of two similar structures. The left structure is a substructure of the right, but the absolute labels are different due to the higher priority oxygen atom.

6.2.4 Comparison

The ability to compare the configuration of stereocentres is essential in determining whether two structures are the same and whether one is a part (substructure) of another.

One way to compare tetrahedral centres could be to match the nonplanar labels directly. Since there are multiple ways nonplanar labels can be used to indicate stereo configuration the approach is not correct. The approach has however proved useful in atom-mapping where stereochemistry may optionally (consider isomerases) match between reactants and products^[123].

Another method to compare configurations is to assign an absolute stereo descriptor. Stereocentres are then considered identical if they have the same label. Using an absolute descriptor such as CIP or a signature^[237], the labels are only the same for identical structures and cannot be used for substructures (Fig. 6.11). In many cases, the absolute descriptors are conserved in reactions but their use incorrectly identifies a stereo inversion in the conversion of Rifamycin W to Rifamycin W-hemiacetal (KEGG:R06992, MetaCyc:RXN-9607).

To compare the local stereochemistry representation of one structure to another a mapping from one set of atoms to another is required. We shall refer to the structures as the query and target. The mapping of atoms must cover all atoms in both stereocentres. In the case of a substructure or identity search the mapping is conveniently represented by a permutation that reorders all the query vertices to be in the order of the target. A partial mapping can also be represented this way with some atoms in the query unmapped.

Recall the tetrahedral data structure consists of a central focus atom, four neighbours and a winding. For the example (Fig. 6.12), two isomorphisms are found from the query to the target atoms. These are represented by two permutations (or renumber-

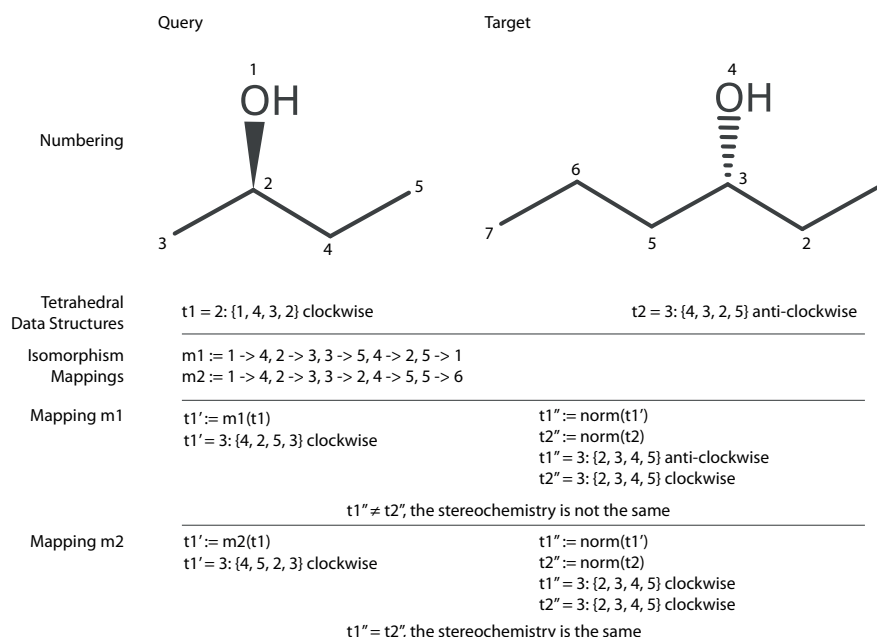


Figure 6.12: Comparing the stereochemistry of a query structure to a target. Given a mapping from the query vertices to the target vertices, the query stereocentre is mapped to new atom ordering. Once normalised the windings/configurations are inspected for equivalence. Only *m2* preserves the stereochemistry.

ings) of the atoms in the query, *m1* and *m2*. A transformed query tetrahedral configuration $t1'$ can be obtained by renumbering its atoms using either mapping. The winding is not modified. Both $t1'$ and $t2$ are then normalised by sorting their neighbours. If an odd number of swaps (permutation parity) was made during the reordering the winding is inverted^[166]. As can be seen in Figure 6.12, only the mapping *m2* preserves the stereochemistry configuration.

Comparison of double bond stereochemistry is accomplished in a similar manner but uses a different normalisation. As only one substituent at each end is stored, each representation may contain different atoms. To account for this the normalisation step exchanges the neighbours in one of the representations to match the other. If only one atom is exchanged, the configuration (opposite or together) is inverted.

This method of comparison can be applied and extended to other stereochemistries by defining how normalisation is carried out.

The comparison of stereochemistry is implemented as a filter and applied after any mappings have been found. In some instances this may be slightly less efficient as

incorrect mappings could have rejected early. The filtering approach allows the enhancement to existing graph mapping algorithms provided in the CDK^[238].

6.2.5 Description

The Cahn-Ingold-Prelog (CIP) labelling provides a method of assigning labels to describe the configuration of stereocentres. The system produces *R/S* and *E/Z* labelling recommended in systematic nomenclature^[239]. As an absolute labelling, assigned labels can be used to compare the stereochemistry of compounds with the same connectivity. Although the labels are not useful for comparing stereochemistry in substructures they are useful for generating, parsing, and describing nomenclature.

The CIP labels are determined by ordering of atoms (referred to as ligands) based on a set of hierarchical rules. For tetrahedral centres, the arrangement is orientated such that lowest rank atom is facing away from the view. Based on rank, the remaining three atoms then proceed either clockwise or anti-clockwise. If neighbours proceed clockwise they are labelled *R* or *S* when they rotate anti-clockwise.

Double bonds are labelled by determining if the highest rank ligand on each side, the ligands are then either together (*Z*) or opposite (*E*).

It should be clear how this is very similar to the data-structures described earlier and that the absolute label provided by the CIP system is a global (absolute) ordering of the stereo representation.

The atoms are ordered by a set of hierarchical sequence rules applied to a directed acyclic graph (digraph). The digraph extends from each centre, creating *ghost* atoms for double bonds and ring closures^[240]. Summarised the rules are:

1. Higher atomic number precedes lower
2. Higher atomic mass precedes lower
3. *Z* precedes *E*
4. Like pair *R,R* or *S,S* precede unlike pairs
5. *R* precedes *S*

The sequence rules have changed over time, for example, rule 3 actually refers to the geometric configuration based on the rank in the digraph and not the absolute



Figure 6.13: Interdependent stereocentres depend on the configuration of one or more other stereocentres. The centre atom (6.13a) is only a stereocentre when the other two configurations are not the same. The other two stereocentres are perceived first and the central atom. Intradependent stereocentres are mutually dependent (6.13b). The configuration of all stereocentres is determined at the same time.

label^[240]. Several others have suggested modifications and extensions to the rules to handle known deficiencies^[239]. This represents a major problem in the labelling system as new rules, that may be incompatible with existing rules, must be added when a new exception is encountered. With the use of different rules, different labels and names for structures may be produced.

The Centres library

The existing CIP implementation in the CDK provided the first two sequence rules for labelling tetrahedral centres^[241]. Labelling of geometric configurations has recently been added. Without using the other higher sequence rules, many metabolites could not be labelled. As the labels were originally used for comparison and hash encoding, stereoisomers of metabolites like inositol (Fig. 6.13b) could not be discriminated.

To investigate the usage and application of the CIP labels, the prototype library *Centres* was created. The library identifies both interdependent and intradependent stereocentres and labels *pseudoasymmetric* descriptors (*r* and *s*) (Fig. 6.13a).

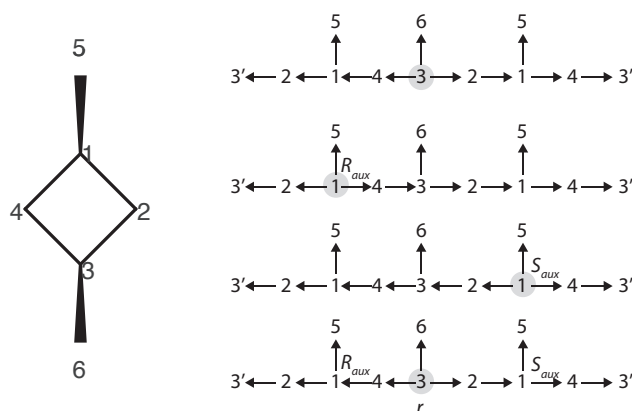


Figure 6.14: Exploring the digraph centre on atom 3, no constitutional differences are seen. The digraph is then re-rooted on the other stereocentre (atom 1) and the configuration computed on the now asymmetric graph. Atom 1 was duplicated and so two auxiliary descriptors are computed. The original diagram can now see a difference and determines that the branch starting at atom 4 has priority.

Auxiliary descriptors In highly symmetric compounds, the created digraph may require auxiliary stereo descriptors. These act to artificially break the symmetries in the digraph by assigning a local stereo label (R_{aux} or S_{aux}) to a node in the digraph (Fig. 6.14). The stereocentre (atom 3) has two constitutionally equivalent neighbours (2 and 4) that cannot be distinguished without the use of auxiliary descriptors. With the completed digraph for atom 3, we re-root the digraph on each node that represents atom 1. Since re-rooted and did not recalculate relative to atom 1 there is now asymmetry and atom 4 or 2 takes priority. When the graph is re-rooted again back on atom 3 we now see that atom 4 has priority over 2 because of rule 5, R precedes S .

6.3 Results and discussion

The five aspects of handling stereochemistry were validated on ChEBI (release 113). Verifying representation and comparison of stereochemistry is isolated and checked by unit tests in the CDK codebase (Apdx. E.1).

6.3.1 Identification

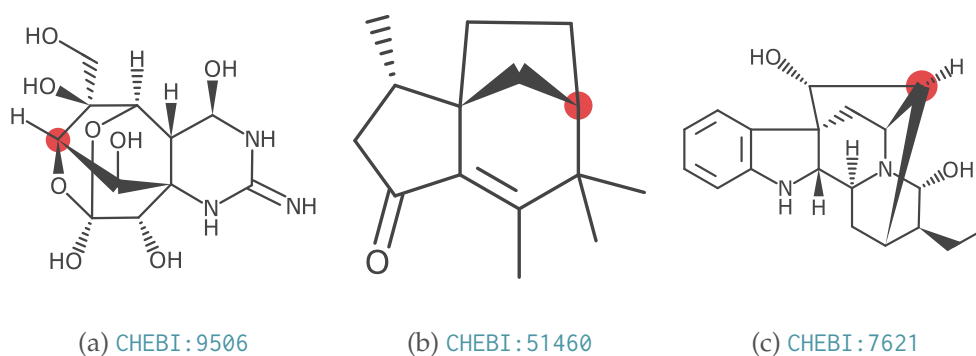


Figure 6.15: Problematic tetrahedral stereocentres. The same nonplanar bonds cannot be next to each other, 6.15a is indicating the configuration using both perspective and nonplanar bonds. The shallow angle of 6.15b is flagged as ambiguous by the InChI. The differences in bond lengths caused the incorrect winding to be calculated, 6.15c.

To ensure stereocentres were being correctly located and assigned, stereochemistry from ChEBI 2D depictions was compared against the InChI. All entries in the ChEBI (release 113) SDfile had an InChI generated first with coordinates (InChI does identification) and then from setting the stereochemistry configuration via the API (CDK does identification). The InChIs were then compared for differences in tetrahedral and geometric assignments.

There were 12,927 entries with at least one tetrahedral centre, 16 (0.12%) entries were found to produce a different InChI when the CDK identification was used (Tab. 6.1). Three entries were flagged as invalid by InChI. These were due to incorrect wedge or hatch placement (Fig. 6.15a). The CDK incorrectly identified these. Another seven entries are identified as ambiguous by the InChI perception due to parallel neighbours (Fig. 6.15b). All these entries were reported to ChEBI and the entries have now been redrawn.

Table 6.1: Perception of tetrahedral centres in CDK and InChI. Structure depictions are listed in Appendix E.2.

ChEBI Entry	Reason for difference
CHEBI:2909	InChI ignored (invalid) – now redrawn in ChEBI
CHEBI:32956	
CHEBI:9506	
CHEBI:29519	InChI ignored (ambiguous) – now redrawn in ChEBI
CHEBI:51460	
CHEBI:51458	
CHEBI:51478	
CHEBI:51479	
CHEBI:51480	
CHEBI:65778	
CHEBI:34596	InChI ignored (extended tetrahedral) – now redrawn in ChEBI
CHEBI:7621	Incorrect winding detected by CDK – now resolved
CHEBI:61677	
CHEBI:63587	
CHEBI:70711	
CHEBI:60095	Skipped by CDK – aromatic bonds

Four entries had the wrong winding assigned by CDK. This was found to be because the determinant calculation was failing for varied bond lengths (Fig. 6.15c). This has now been resolved, the correct configuration is now obtained by normalising bond lengths to unit vectors before determining the winding.

Finally one entry was skipped due to the presence of aromatic bonds in the molfile representation. The symmetry detection skips this compound as it requires Kekulisation and at the time of evaluation it was not performed automatically. An extended tetrahedral stereocentre was missed by the InChI as it did not have enough nonplanar bonds labelled but was correctly identified by CDK.

The ChEBI release contains 21,559 entries that had one or more geometric configuration. The initial computation found 294 (1.36%) entries that had differences in geometric configuration. The majority (292) were due to the disabled ability to identify geometric centres in rings. These stereocentres do have a configuration but the identification excludes all cyclic double bonds to avoid problems with the layout.

One of the remaining incorrect perceptions was due to the CDK treating wavy (up or down) bonds as bidirectional and so excluding the double configuration in tefluthrin ([CHEBI:9430](#)). The tefluthrin entry has now been modified by the ChEBI curators (Apdx. E.3). The remaining entry was the compound also skipped in tetrahedral differences ([CHEBI:60095](#)) due to aromatic bonds.

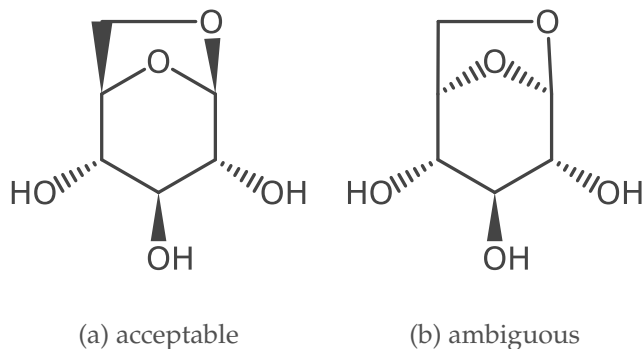


Figure 6.16: InChI perception of stereochemistry in [ChEBI:30997](#) depends on the placement of nonplanar labels.

6.3.2 Depiction

The InChI was again used in the validation of depictions. For correctness, the CDK library should be able to generate a depiction that obtains the same InChI as the original depiction.

For tetrahedral centres, an InChI was calculated with the original depiction in ChEBI. Nonplanar bonds were then reassigned (coordinates stay the same) and another InChI calculated. Reassigning nonplanar labels caused 10 differences in the generated InChI identifier. Five of these entries were due to incorrect identification discovered previously. The other five were entries with a shallow angle that were flagged as ambiguous only after the nonplanar bonds had been reassigned (Fig. 6.16).

To evaluate geometric isomers, coordinates are regenerated caused 178 (0.84%) geometric stereocentres to be incorrectly depicted. These were again primarily in macrocycles. This number is less than the differences found during identification because the generated diagram may have unintentionally been assigned the correct configuration.

6.3.3 Description

To validate and investigate differences in absolute stereochemistry labelling (CIP), labels assigned by CDK 1.4 and *Centres* were compared to JChem (ChemAxon).

When comparing CDK 1.4 to JChem, a total of 78,246 tetrahedral centres were labelled by either program. 3,779 (4.8%) labels were found to be different and 564 (0.72%) were

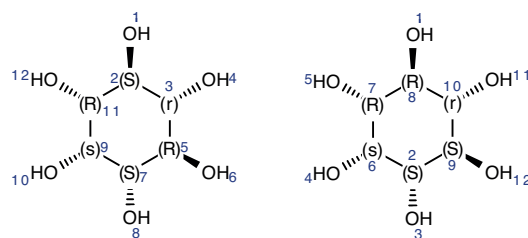


Figure 6.17: CIP labelling for myo-inositol in MarvinSketch (JChem) with different atom order. The absolute labels change based on atom order.

unlabelled by one or the other. The majority of missed labels were due to CDK only supporting the first two sequence rules. The labelling differences appeared to be due to the CDK implementation comparing the digraphs in a depth-first and not breath-first-traversal.

When comparing *Centres* to JChem, 78,120 tetrahedral centres were labelled by either tool. The small difference in number was because *Centres* timed out for 10 entries. Of the 78,120 labels, 126 (0.16%) were in disagreement and 24 (0.03%) were unlabelled by JChem.

The first 30 structures with different labels were manually inspected and compared with additional CIP implementations (Tab. 6.2). In the first 30 entries there were 36 tetrahedral centres that *Centres* and JChem labelled differently. No implementations were found to agree completely on the labelling of these structures (Tab. 6.3). *Centres* and ChemSketch were most similar finding the same label for 27/36 atoms. CDK and Accelrys Draw also agreed relatively closely as many centres were not assigned a label. RDKit and JChem agreed on 21/36 labels.

The RDKit implementation is based on extended connectivity^[242] rather than acyclic hierarchical digraphs. The method includes a slight modification to the rules that attempts to resolve an ambiguity due to multiple resonance forms. This difference could be why different labels are found. The similarity with the JChem labels suggest the same algorithm may be partially used. The algorithm is not capable of labelling intradependent stereocentres which accounts for the missing labels in RDKit. JChem does label intradependent stereocentres but the labelling was found to be inconsistent and varied under reordering (Fig. 6.17). As an absolute label, the value should be the same regardless of the atom ordering.

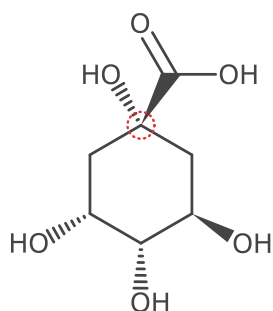
In the ChEBI entries, 24 tetrahedral centres were not labelled by JChem; 14 of these labels were seen to be interdependent configurations (Fig. 6.18a). The remaining 10

Table 6.2: Labels of tetrahedral centres for entries in ChEBI. Structure depictions are available in Appendix E.4.

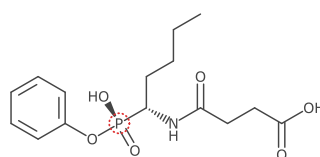
ChEBI Entry	Atom	Centres	CDK 1.4	JChem	RDKit	Accelrys Draw	ChemSketch
CHEBI:3048	2	<i>r</i>	<i>R</i>	<i>R</i>	<i>R</i>	-	<i>r</i>
CHEBI:15742	3	<i>s</i>	-	<i>S</i>	<i>S</i>	-	<i>s</i>
CHEBI:15884	3	<i>r</i>	-	<i>R</i>	<i>R</i>	-	<i>r</i>
CHEBI:16063	35	<i>S</i>	<i>R</i>	<i>R</i>	<i>R</i>	<i>S</i>	<i>S</i>
CHEBI:16086	26	<i>R</i>	-	<i>S</i>	<i>S</i>	-	<i>S</i>
	59	<i>R</i>	-	<i>S</i>	<i>S</i>	-	<i>S</i>
CHEBI:16226	10	<i>S</i>	<i>R</i>	<i>R</i>	<i>R</i>	<i>S</i>	<i>S</i>
CHEBI:16419	11	<i>S</i>	<i>R</i>	<i>R</i>	<i>R</i>	<i>S</i>	<i>S</i>
CHEBI:16684	19	<i>r</i>	<i>R</i>	<i>S</i>	<i>R</i>	-	<i>r</i>
CHEBI:16794	10	<i>s</i>	<i>R</i>	<i>S</i>	<i>S</i>	-	<i>s</i>
CHEBI:17268	4	<i>R</i>	-	<i>S</i>	-	-	<i>R</i>
	6	<i>S</i>	-	<i>R</i>	-	-	<i>S</i>
CHEBI:17401	3	<i>s</i>	-	<i>S</i>	-	-	<i>s</i>
CHEBI:17486	19	<i>r</i>	<i>R</i>	<i>S</i>	<i>R</i>	-	<i>r</i>
CHEBI:17521	1	<i>s</i>	-	<i>S</i>	-	-	<i>S</i>
CHEBI:18013	1	<i>s</i>	-	<i>S</i>	-	-	<i>S</i>
CHEBI:18173	3	<i>r</i>	-	<i>R</i>	-	-	<i>R</i>
CHEBI:27831	15	<i>R</i>	<i>R</i>	<i>S</i>	<i>S</i>	<i>R</i>	<i>R</i>
CHEBI:27987	3	<i>r</i>	-	<i>R</i>	-	-	<i>r</i>
	4	<i>r</i>	-	<i>S</i>	-	-	<i>r</i>
CHEBI:28271	15	<i>R</i>	<i>S</i>	<i>S</i>	<i>S</i>	<i>R</i>	<i>R</i>
CHEBI:28332	33	<i>R</i>	<i>S</i>	<i>S</i>	<i>S</i>	<i>R</i>	<i>R</i>
CHEBI:29751	1	<i>s</i>	-	<i>S</i>	<i>R</i>	-	<i>s</i>
CHEBI:30164	2	<i>r</i>	<i>R</i>	<i>s</i>	<i>R</i>	-	<i>r</i>
CHEBI:50692	12	<i>R</i>	<i>R</i>	<i>S</i>	<i>S</i>	<i>S</i>	<i>S</i>
CHEBI:65954	6	<i>S</i>	<i>R</i>	<i>R</i>	<i>R</i>	<i>S</i>	<i>S</i>
CHEBI:553542	19	<i>r</i>	-	<i>R</i>	<i>R</i>	-	<i>r</i>
	25	<i>r</i>	-	<i>R</i>	<i>R</i>	-	<i>r</i>
CHEBI:512	3	<i>s</i>	-	<i>S</i>	<i>R</i>	-	-
CHEBI:2909	22	<i>R</i>	-	<i>S</i>	<i>S</i>	<i>R</i>	-
CHEBI:3049	2	<i>r</i>	<i>S</i>	<i>R</i>	<i>R</i>	-	<i>r</i>
CHEBI:5956	19	<i>r</i>	-	<i>R</i>	<i>R</i>	-	<i>r</i>
	25	<i>r</i>	-	<i>R</i>	<i>S</i>	-	<i>r</i>
CHEBI:5957	16	<i>r</i>	-	<i>R</i>	<i>R</i>	-	<i>r</i>
	22	<i>r</i>	-	<i>R</i>	<i>R</i>	-	<i>r</i>
CHEBI:7621	16	<i>S</i>	<i>R</i>	<i>R</i>	<i>S</i>	<i>R</i>	-

Table 6.3: Agreement between CIP labels assigned by different programs. The number indicates the number of tetrahedral stereocentre labels (from Tab. 6.2) in which the each tool agreed (total = 36). A higher number indicates the tools agree with the labelling.

	<i>Centres</i>	CDK 1.4	JChem	RDKit	Accelrys Draw	ChemSketch
<i>Centres</i>	-	2	0	1	8	27
CDK 1.4		-	8	18	22	3
JChem			-	21	2	6
RDKit				-	9	3
Accelrys Draw					-	9
ChemSketch						-



(a) CHEBI:17521



(b) CHEBI:43012

Figure 6.18: Tetrahedral centres that are labelled by *Centres* but ignored by JChem. The first is (6.18a)) a interdependent stereocentre. The second (6.18b) is ignored due to the mobile hydrogen in JChem (and CDK) but not *Centres*.

involved potentially non-stereogenic phosphorus and nitrogen atoms (Fig. 6.18b) and ambiguous nonplanar bond placement (Fig. 6.15a). The InChI would not indicate a configuration of the phosphorus in CHEBI:43012 (Fig. 6.18b) due to the mobile hydrogen. As *Centres* aims to only provide a CIP label and not normalise representations, these atoms were assigned a label.

6.4 Conclusions

Stereochemistry in the Chemistry Development Kit has been improved and can now be efficiently and accurately handled. The configuration of stereocentres can be retrieved, stored, identified, displayed and compared between multiple representations and formats. One missing computation is the generation of 3D coordinates with the correct spatial arrangement. This functionality was not included as support for 3D coordinate generation in the CDK is currently limited.

The identification and depiction of stereochemistry has scope for improvement. In particular, stereocentres that are ambiguous have a configuration assigned when they should be flagged as problematic. The depiction should also ensure that ambiguous depictions are not generated. This involves either adding hydrogens, or selecting a more appropriate nonplanar bonds assignment. Depicting geometric isomers of macrocycles is required but requires significant changes to the structure diagram layout.

The identification of stereochemistry in 2D depictions relies on the use of nonplanar bonds. As was seen in Chapter 2, many structures (particularly carbohydrates) can be drawn in a perspective projection such as Fischer or Haworth. Identification of the stereochemistry is possible with specialised handling^[243]. This has not been included as the two resources in which these depictions were encountered (KEGG and Meta-Cyc) have been updated and had these structures redrawn. However, it would be useful to provide this functionality.

The location of asymmetries and classification of stereocentres could also be improved. A more comprehensive algorithm based on group theory and the inspection of automorphisms^[162] offers a comprehensive solution. An algorithm for generating automorphism was recently added to the CDK by Gilleain Torrance^[244] and its use is being explored for improved stereo detection.

The current stereochemistry data structures only distinguish between discrete configurations. Explicitly indicating whether a stereocentre is *unknown* (i.e. either configuration) or *undefined* would be of use in structure queries and normalisation. An unspecified tetrahedral stereocentres are indicated by a wavy (up or down) bond whilst multiple conventions are used for double bonds^[245]. An additional *ambiguous* configuration could also be used to mark problematic stereocentres identified in a depiction. This would improve the conversion accuracy to the InChI.

To our knowledge, the CDK is currently the only open cheminformatics library to completely handle extended tetrahedral stereochemistry and convert between SMILES, InChI, and depictions (inc. molfile). Additional data structures to represent square planar, trigonal bipyramidal, and octahedral stereochemistry could be added. There are a small number of structures with these stereochemistries in ChEBI and ChEMBL^[246]. Including support for these stereochemistries would be a small but important benefit.

The comparison of stereocentres has been used to enrich isomorphism and substructure searching. SMARTS queries include logical stereochemistry predicates that are now supported.

The CIP labelling library, *Centres*, has been utilised in the EC-BLAST tool for comparing enzymatic reactions^[122]. The labels are used in the atom-atom mapping within reactions. Although in many cases the stereochemistry of a CIP label is preserved, the EC-BLAST would benefit from using the generic local stereochemistry comparison.

The *Centres* library is able to assign labels that agree more closely with other toolkits than CDK 1.4. To our knowledge *Centres* is the only open source and freely available implementation that is capable of naming intradependent stereocentres. In future the *Centres* implementation will be integrated in the CDK.

The nature of the CIP systems means a subtle difference in implementation or rule application can produce a different label. As the CIP labels are the primary means of communicating stereochemistry in nomenclature, it is interesting to see that there are differences even for some simple cases. Although small, the differences highlight the need for a reference implementation that could be used to verify the correct labelling.

6.5 Availability

The methods and implementations discussed in this chapter are openly available in the Chemistry Development Kit, <http://cdk.github.io/cdk>. The Java class names are:

- `org.openscience.cdk.stereo.StereoElementFactory` – creation of stereochemistry from 2D and 3D depictions
- `org.openscience.cdk.stereo.Stereocenters` – location and validation of stereochemistry
- `org.openscience.cdk.layout.NonplanarBonds` – assignment of nonplanar labels to an existing depiction

-
- [org.openscience.cdk.layout.CorrectGeometricConfiguration](#) – correcting layout of geometric isomers
 - [org.openscience.cdk.isomorphism.StereoMatch](#) – check stereochemistry configuration in sub-structure and identity searches
 - [org.openscience.cdk.isomorphism.SmartsStereoMatch](#) – check stereochemistry configuration in SMARTS queries

The assignment of Cahn-Ingold-Prelog (CIP) labels has not yet been integrated but is available through the *Centres* library, <http://johnmay.github.io/centres>

Chapter 7

Conclusions

Genome-scale metabolic reconstructions are an important resource for studying metabolism. The detection of common components between reconstruction is beneficial to their creation and study.

In Chapter 2, a technique for the alignment of reconstructions based purely on chemical structure annotations was introduced. Through the graph theoretic representation of chemical structures, identical and similar metabolites could be correctly matched and categorised.

Comparing the alignment to existing mappings from a merged database (MNXref) and a published pairwise mapping highlighted some differences. Differences were primarily due to inconsistent and erroneous annotations found both in databases and published reconstructions. The databases were notified of the discovered inconsistencies, improving their quality.

Using the technique, newly discovered matches were found both with a one-to-one or one-to-many mapping. This is an improved alignment over those used for validation and is important for procedures such as gap filling. In gap filling, an improved alignment means more candidate reactions are available for resolving dead-end metabolites. The one-to-many mappings represent the potential to split or merge metabolite entries for a more accurate alignment and reconstruction.

Aligning reconstructions in this way requires precise annotation of metabolites. The Metingear desktop application, described in Chapter 3, was created to import, manage, modify, and annotate genome-scale reconstructions. The application was used to successfully import and export reconstructions that were used in Chapter 2. Precise

metabolite annotation was made possible through automated, semi-automated, and manual curation. External resources are accessed dynamically allowing the tool to operate independently of data availability.

Algorithms for finding cycles in graphs (chemical rings) were described in Chapter 4. These algorithms are fundamental to many other methods and utilised in: hash codes, atom typing, stereochemistry, aromaticity, fingerprints, and descriptor calculations. With the efficient and fast implementations, processing that would previously take minutes can now be accomplished in seconds. This has benefits to other projects utilising the library, not just in metabolism but in the wider chemistry community.

Procedures for delocalising and Kekulising structures explored in Chapter 5 have been used extensively for input and output. As a bonus, the efficient Kekulisation could be repurposed and used to efficiently generate tautomers.

The handling of stereochemistry in the CDK was unified and improved in Chapter 6. There are still areas to address, mainly in perception and depiction. These challenges are not unique to the CDK. With perception, the crux of the problem is the inability to explicitly define stereocenter locations in 2D depiction formats (e.g. CTab V2000). The CTab V3000 and CML formats resolve this issue but have limited adoption.

The observed variation in labelling tetrahedral centres with Cahn-Ingold-Prelog (CIP) descriptors was surprising. As these descriptors are the primary way of described configurations in nomenclature, a more in-depth analysis would be useful to quantifying the differences and standardising labelling.

The methodologies described in this thesis open up several new lines of investigation. Annotating metabolites with structure representations provides a chemically relevant network for a specific reconstruction.

Of particular interest is the inspection of metabolite similarity and transformations. This allows one to assess whether two metabolic reconstructions are performing similar chemical transformations and if different classes of metabolites are synthesised. The transformations can also be utilised to characterise novel pathways and potentially close network gaps.

Through inspecting the transformations and gaps in a network, there is potential to identify promiscuous enzymes that close these gaps. These reactions produce metabolite structures that may be present or absent in the existing network. To identify these, the alignment described in Chapter 2 can be utilised.

Future work

Assisted alignment

The focus of the reconstruction alignment was on a single pass matching. Performing an iterative matching that considers other pairings and reaction contexts could improve the alignment and resolve match conflicts.

Correct and incorrect candidates were retrieved as diastereoisomers and structurally related. Refining the alignment to keep the correct matches may be possible through inspecting the reaction context. A more robust approach would be to allow user confirmation and could prompt when to split or merge metabolites. To allow for this, integration with Metingear is being explored.

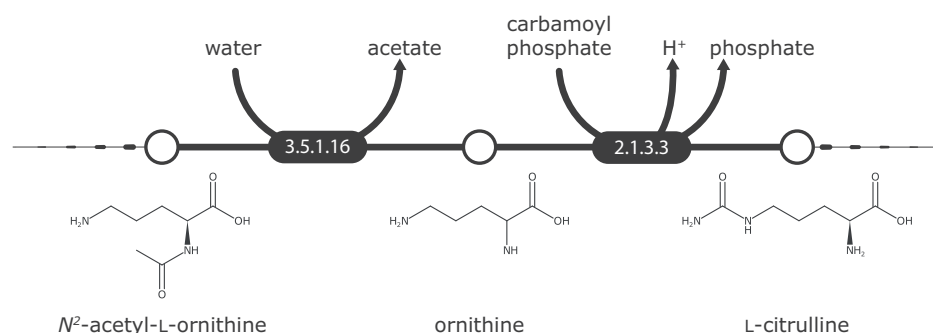
Error detection and correction

The annotation quality of databases and reconstructions is continuously improving but better detection of inconsistencies is needed.

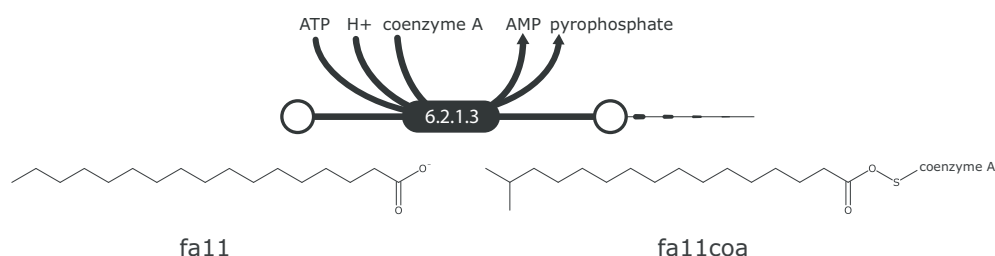
Validating chemical structure: formula, stereochemistry, and valence can be an indication of a mistake or missing information^[144]. This however only considers local information about the metabolite. The inspection of network associations has proved useful in detecting metabolic genes^[247] and propagating annotations^[151]. In a similar manner, it may be possible to detect and potentially even resolve inconsistencies by inspecting the chemical transformations.

In the *iJR904* reconstruction, the metabolite ornithine ([orn](#)) identified the L and D enantiomers in MetaCyc. In reality, the naming of ornithine in the reconstruction is ambiguous and was intended to represent L-ornithine.

To a human, it may be obvious that the left-handed and more common amino acid was intended. This is more difficult to reason about computationally. However, by inspecting the reactions of the metabolite, the mistake could be identified:



The same is true of the erroneous fatty acid structures in *iBsu1103* and SEED:



With the chemical structures available, a reaction can be described by the transformation. These reactions have an EC number and the expected transformation pattern could be verified. With a pre-defined pattern, stereochemistry and bonding could be corrected automatically. When no EC number is assigned, calculating an atom-atom mapping would locate two *reaction centres* on a single structure. This may indicate a multi-step reaction but these reactions could be flagged for inspection.

Unambiguous representation

The key to using the structure is that it provides an unambiguous description of a metabolite. Entries from *iJR904*, *iYO844*, and *iBsu1103* that could not have an unambiguous structure assigned were: acyl-carrier-protein, thioredoxin, tRNA attached amino acids, unsaturated fatty acids, and polysaccharides. Virtual metabolites are also problematic but are specific to each reconstruction.

For the non-virtual metabolites, there are two problems: macromolecules and generic structures.

Macromolecules can be represented in full atomic detail but this is often inefficient and

inappropriate. However, more appropriate representations (e.g. peptide sequence) can fail to capture subtleties such as modifications and attachments. Efforts such as the Hierarchical Editing Language for Macromolecules (HELM)^[248] attempt to bridge this gap but have limited software support and use.

Generic metabolites attempt to represent more than one discrete entity. These include, defined Markush structures with variable Rgroups but a common scaffold or mixtures with variable bonding (polysaccharides and unsaturated fatty acids). Markush structures can be effectively represented but mixtures are more difficult. The molfile Sgroups provide a means to describe mixtures with specified data labels but this is little improvement over naming (a consensus data label must be used).

For some classes of compound, if feasible, it may be more appropriate to enumerate the generic reactions. This was actually required to ensure correct mass balance in Recon 2^[249].

Metingear outlook

Metingear was developed during the course of this project and features were integrated as and when they were needed. Some improvements of the CDK have been added but there are many more that would be useful, including additional standardisation operations.

The user interface was designed iteratively and a retrospective view highlights some layout optimisations that would improve editing. Two longer term goals are the round tripping of all annotations to the SBML standard and the inclusion of more external resources (e.g. Rhea).

CDK outlook

Methods discussed in this thesis required many improved algorithms and maintenance to the CDK library. The library is now far more capable, robust, and efficient but requires ongoing support and development. A recent key step was the conversion to use the Maven build tool^[250]. This simplifies the distribution of releases and facilitates the use of the CDK in other projects.

A limiting factor in the performance is now the conversion to an optimised data structure. Using these by default is hindered by the size and maturity of the existing codebase (1,200 source files). As the size of the codebase is difficult to maintain and streamlining the focus and scope of API on key functionality is needed.

Final remarks

All methods and software implementations discussed in this thesis have been openly developed and released. It has sometimes been a challenge to overcome the limitations this has imposed. Through contributing these to the community, it is hoped they are found useful and extended to related metabolic research and other areas.

APPENDICES

Appendix A

Chapter 2 supplementary material

A.1 InChI API differences

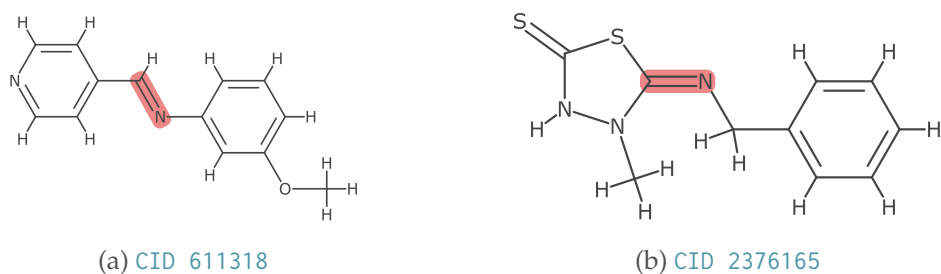


Figure A.1: Geometric isomers perceived by the InChI but not encoded in the InChI attached to the PubChem entry.

The InChI and InChIKey annotated in PubChem entries are computed from an internal format and are different from those generated with the molfile and binary (Fig. A.1). When the InChI toolkit is used through the API, the stereochemistry of elements may be optionally specified. When invoked on a depiction, the InChI algorithm may include (or exclude) stereocentres that another toolkit would not provide. As the hash code was being tested on the molfile input, the InChIKey of every entry was recalculated from this representation. In total, 240746 entries (~0.5%) had different InChIKeys when recomputed.

A.2 Description of cycle notation

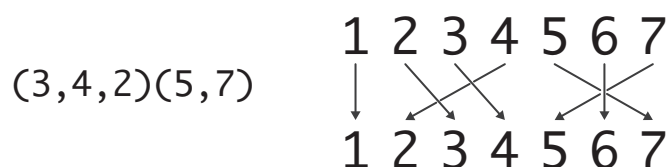


Figure A.2: Cycle notation is a concise way of representing a permutation. Any element that does not move (1 and 6) is omitted. The cycles denote each element is replaced with that to its right. For, (3,4,2), 3 goes to 4, 4 goes to 2, and 2 goes to 3.

A.3 Guide to database identifiers

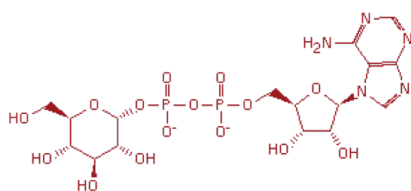
This and subsequent chapters reference several external databases. This section provides details on what each identifier is and where it can be located. Identifiers are highlighted and all except BiGG are hyperlinked when read in digital form. Direct links via the MIRIAM registry are:

Resource	Example	Direct lookup
BiGG	orn	-
ChEBI	CHEBI:27931	http://identifiers.org/chebi/CHEBI:27931
KEGG COMPOUND	C05898	http://identifiers.org/kegg.compound/C05898
MetaCyc	L-FUCOSE	http://identifiers.org/biocyc/META:L-FUCOSE
PubChem-Compound	CID 4643229	http://identifiers.org/pubchem.compound/4643229
SEED	cpd00870	http://seed-viewer.theseed.org/seedviewer.cgi? page=CompoundViewer&compound=cpd00870

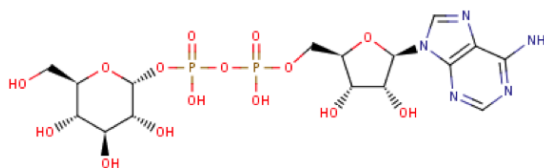
The *iJR904* and *iYO844* were both constructed by the System Biology Research Group at UCSD: <http://systemsbiology.ucsd.edu/Researchers/Palsson>. The reconstructions, were at least partially build in SimPheny and use proprietary abbreviations to identify entries. The abbreviations can be searched in the BiGG database, <http://bigg.ucsd.edu/>.

A.4 MetaCyc structure corrections

The exploratory analysis identified incorrect structure representations in MetaCyc. These entries were ADP- α -D-glucose, that had the wrong attachment to the adenine and β -L-fucose 1-phosphate that depicted the α form.



(a) ADP- α -D-glucose (ADP-D-GLUCOSE)



(b) ADP- α -D-glucose (CHEBI:15751)

A.5 Formula key implementation

A formula key encodes the element frequencies of a structure. Hydrogens can optionally be excluded. The frequencies are encoded using key-value counting (Listing A.1).

Listing A.1: Key-value counting of element frequencies in Java

```
long encode(IAtomContainer cmpd, boolean h) {  
  
    int[] count = new int[118];  
    for (IAtom atm : cmpd.atoms()) {  
        count[elemNum(atm)] += 1;  
        count[1] += labelledHCount(atm);  
    }  
  
    long hash = (1L << 31) - 1;  
    for (int i = h ? 1 : 2; i < count.length; i++) {  
        hash = 31 * (hash + count[i]);  
    }  
  
    return hash;  
}
```

A.6 Problematic tetrapyrrole representations

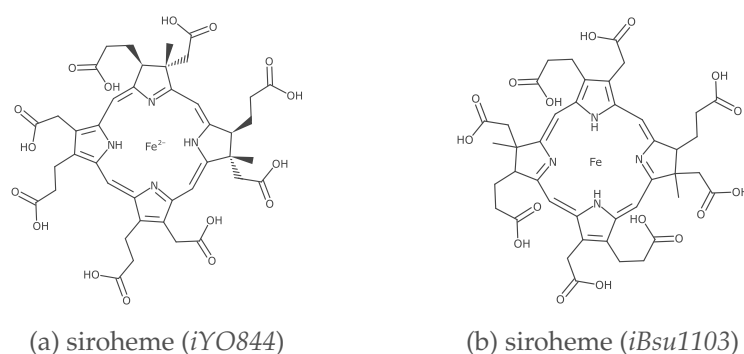


Figure A.4: Tetrapyrrole derived coordination complexes such as porphyrin and corrin are precursors to important cofactors, metabolites and vitamins. Their structural representations present a challenge to match with structural identity. Firstly, the coordination bonds to the metal may be disconnected or connected using covalent bonds or query 'any' bonds. Secondly, the nitrogen in each pyrrole may be substituted or protonated. There are four of these nitrogens and so these hydrogens may optionally be present and in different combinations. Thirdly, representations may include only the sulphur or full cysteine residues from the protein to which the complex is bound. Finally, the substituents at the periphery of complex be attached in different combinations.

A.7 Alignment of *iJR904* and MetaCyc 17.5

A.7.1 Without structure annotations

<i>Macromolecule / polysaccharide</i>	
Abrv	Name (<i>iJR904</i>)
ACP	acyl carrier protein
trdox	Oxidized thioredoxin
trdrd	Reduced thioredoxin
apoACP	apoprotein [ACP]
glycogen	glycogen

<i>Transfer RNA</i>	
Abrv	Name (<i>iJR904</i>)
glutna	L-Glutamyl-tRNA(Glu)
trnaglu	tRNA (Glu)

<i>Virtual</i>	
Abrv	Name (<i>iJR904</i>)
12dgr_EC	1,2-Diacylglycerol (E.coli) **
agpc_EC	acyl-glycerophosphocholine (E.coli) **
agpe_EC	acyl-glycerophosphoethanolamine (E.coli) **
agpg_EC	acyl-glycerophosphoglycerol (E.coli) **
apg_EC	acyl phosphatidylglycerol (E.coli) **
cdpdag1	CDPdiacylglycerol (E coli) **
clpn_EC	Cardiolipin (Ecoli) **
eca_EC	Enterobacterial common antigen polysaccharide (Ecoli)
lps_EC	lipopolysaccharide (Ecoli)
pa_EC	phosphatidate (E.coli) **
pc_EC	Phosphatidylcholine (E.coli) **
pe_EC	Phosphatidylethanolamine (Ecoli) **
peptido_EC	Peptidoglycan subunit of Escherichia coli
pg_EC	Phosphatidylglycerol (Ecoli) **
pgp_EC	Phosphatidylglycerophosphate (Ecoli) **
ps_EC	phosphatidylserine (Ecoli) **

A.7.2 Stereoisomer candidates

- FIC: Fixed in ChEBI
- FIM: Fixed in MetaCyc

Less specific (single)

Abrv	Name (<i>iJR904</i>)	Retrieved	Comment
------	------------------------	-----------	---------

23dhdp	2,3-Dihydrodipicolinate	2-3-DIHYDRODIPICOLINATE	<i>R/S</i>
pan4p	Pantetheine 4'-phosphate	PANTETHEINE-P	<i>R/S</i>
thdp	2,3,4,5-Tetrahydrodipicolinate	DELTA1-PIPERIDEINE-2-6-DICARBOXYLATE	<i>R/S</i>
2dr1p	2-Deoxy-D-ribose 1-phosphate	DEOXY-D-RIBOSE-1-PHOSPHATE	α/β
5mdr1p	5-Methylthio-5-deoxy-D-ribose 1-phosphate	CPD-444	α/β
acgam1p	N-Acetyl-D-glucosamine 1-phosphate	N-ACETYL-D-GLUCOSAMINE-1-P	α/β
adphep-D,D	ADP-D-glycero-D-manno-heptose	ADP-D-GLYCERO-D-MANNO-HEPTOSE	α/β
adphep-L,D	ADP-L-glycero-D-manno-heptose	ADP-L-GLYCERO-D-MANNO-HEPTOSE	α/β
air	5-amino-1-(5-phospho-D-ribosyl)imidazole	5-PHOSPHORIBOSYL-5-AMINOIMIDAZOLE	α/β
dt dp4aaddg	dTDP-4-acetamido-4,6-dideoxy-D-galactose	TDP-FUC4NAC	α/β
dt dp4addg	dTDP-4-amino-4,6-dideoxy-D-glucose	CPD-472	α/β
dt dp4d6dm	dTDP-4-dehydro-6-deoxy-L-mannose	DTDP-DEOH-DEOXY-MANNOSE	α/β
dt dprmn	dTDP-L-rhamnose	DTDP-RHAMNOSE	α/β
flp	D-Fructose 1-phosphate	FRU1P	α/β
fdp	D-Fructose 1,6-bisphosphate	FRUCTOSE-16-DIPHOSPHATE	α/β
fgam	N2-Formyl-N1-(5-phospho-D-ribosyl)glycinamide	5-P-RIBOSYL-N-FORMYLGLYCINEAMIDE	α/β
fpram	2-(Formamido)-N1-(5-phospho-D-ribosyl)acetamidine	5-PHOSPHORIBOSYL-N-FORMYLGLYCINEAMIDINE	α/β
fuc1p-L	L-Fucose 1-phosphate	CPD-488	α/β
gar	N1-(5-Phospho-D-ribosyl)glycinamide	5-PHOSPHO-RIBOSYL-GLYCINEAMIDE	α/β
lcts	Lactose	LACTOSE	α/β
man1p	D-Mannose 1-phosphate	MANNOSE-1P	α/β
melib	Melibiose	MELIBIOSE	α/β
prbatp	1-(5-Phosphoribosyl)-ATP	PHOSPHORIBOSYL-ATP	α/β
uaccg	UDP-N-acetyl-3-O-(1-carboxyvinyl)-D-glucosamine	UDP-ACETYL-CARBOXYVINYL-GLUCOSAMINE	α/β
uacmam	UDP-N-acetyl-D-mannosamine	UDP-MANNAC	α/β
uama	UDP-N-acetylmuramoyl-L-alanine	CPD0-1456	α/β
uamag	UDP-N-acetylmuramoyl-L-alanyl-D-glutamate	UDP-AA-GLUTAMATE	α/β
uamr	UDP-N-acetylmuramate	UDP-N-ACETYLMURAMATE	α/β
udpgal	UDPGalactose	CPD-14553	α/β
udpgalfur	UDP-D-galacto-1,4-furanose	UDP-D-GALACTO-14-FURANOSE	α/β
unaga	Undecaprenyl diphospho N-acetyl-glucosamine	ACETYL-D-GLUCOSAMINYLDIPHOSPHO-UNDECAPRE	α/β

<i>Less specific</i>			
Abrv	Name (<i>iJR904</i>)	Retrieved	Comment
3hcinnm	3-hydroxycinnamic acid	CPD-10797	
3ig3p	C'-(3-Indolyl)-glycerol 3-phosphate	INDOLE-3-GLYCEROL-P	
4r5au	4-(1-D-Ribitylamino)-5-aminouracil	AMINO-RIBOSYLAMINO-1H-3H-PYR-DIONE	
hkndd	2-Hydroxy-6-oxonona-2,4-diene-1,9-dioate	CPD-157	
hkntd	2-hydroxy-6-ketononatrienedioate	CPD0-2184	
kdo2lipid4	KDO(2)-lipid IV(A)	KDO2-LIPID-IVA	
kdo2lipid4L	KDO(2)-lipid IV(A) with laurate	KDO2-LAUROYL-LIPID-IVA	
pphen	Prephenate	PREPHENATE	
prlp	5-[(5-phospho-1-deoxyribulos-1-ylamino)methylideneamino]-1-(5-phosphoribosyl)imidazole-4-carboxamide	PHOSPHORIBULOSYL-FORMIMINO-AICAR-P	
u23ga	UDP-2,3-bis(3-hydroxytetradecanoyl)glucosamine	OH-MYRISTOYL	
u3hga	UDP-3-O-(3-hydroxytetradecanoyl)-D-glucosamine	UDP-OHMYR-GLUCOSAMINE	

<i>Diastereomer</i>			
Abvr	Name (iJR904)	Retrieved	Comment
5aprbu	5-Amino-6-(5'-phosphoribitylamino)uracil	CPD-1086	FIM
5prdbz	N1-(5-Phospho-alpha-D-ribose)-5,6-dimethylbenzimidazole	ALPHA-RIBAZOLE-5-P	FIM
ckdo	CMP-3-deoxy-D-manno-octulosonate	CMP-KDO	
u3aga	UDP-3-O-(3-hydroxytetradecanoyl)-N-acetylglucosamine	UDP-OHMYR-ACETYLGUCOSAMINE	
ugmd	UDP-N-acetylmuramoyl-L-alanyl-D-gamma-glutamyl-meso-2,6-diaminopimelate	UDP-AAGM-DIAMINOHEPTANEDIOATE	FIC
<i>More specific</i>			
Abvr	Name (iJR904)	Retrieved	Comment
10fthf	10-Formyltetrahydrofolate	10-FORMYL-THF	
23dhmp	(R)-2,3-Dihydroxy-3-methylpentanoate	1-KETO-2-METHYLVALERATE	
2ohph	2-Octaprenyl-6-hydroxyphenol	2-OCTAPRENYL-6-HYDROXYPHENOL	
2omph	2-Octaprenyl-6-methoxyphenol	2-OCTAPRENYL-6-METHOXYPHENOL	
3c4mop	3-Carboxy-4-methyl-2-oxopentanoate	CPD-7100	
4per	4-Phospho-D-erythronate	ERYTHRONATE-4P	
5mthf	5-Methyltetrahydrofolate	5-METHYL-THF	
aacoa	Acetoacetyl-CoA	ACETOACETYL-COA	
bbtcoa	gamma-butyrobetainyl-CoA	GAMMA-BUTYROBETAINYL-COA	
ctbtcoa	crotonobetainyl-CoA	CROTONOBETAINYL-COA	
dmlz	6,7-Dimethyl-8-(1-D-ribityl)lumazine	DIMETHYL-D-RIBITYL-LUMAZINE	
g3pe	sn-Glycero-3-phosphoethanolamine	L-1-GLYCEROPHOSPHORYLETHANOL-AMINE	
g3pi	sn-Glycero-3-phospho-1-inositol	CPD-541	
gtspmd	Glutathionylspermidine	GLUTATHIONYLSPERMIDINE	
idp	IDP	IDP	
itp	ITP	ITP	
lgt-S	(R)-S-Lactoylglutathione	S-LACTOYL-GLUTATHIONE	
malcoa	Malonyl-CoA	MALONYL-COA	
ohpb	2-Oxo-3-hydroxy-4-phosphobutanoate	3OH-4P-OH-ALPHA-KETOBUTYRATE	
pmcoa	Pimeloyl-CoA	CPD-558	
rdmbzi	N1-(alpha-D-ribose)-5,6-dimethylbenzimidazole	ALPHA-RIBAZOLE	
ribflv	Riboflavin	RIBOFLAVIN	
thf	5,6,7,8-Tetrahydrofolate	THF	
<i>Unspecific</i>			
Abvr	Name (iJR904)	Retrieved	Comment
lipa	KDO(2)-lipid (A)	KDO2-LIPID-A	

A.7.3 Structurally related candidates

- SNP: Correct structure not provided in MetaCyc
- RED: Retrieval deficiency, correct structure had missing or wrong stereo

- TTP: The structures are tetrapyrroles and have differences in representation

<i>Skeleton</i>				
Abrv	Name (iJR904)	Retrieved	Comment	
ttdcea	tetradecenoate (n-C14:1)	CPD-7836	no	SNP
dhptd	4,5-dihydroxy-2,3-pentanedione	CPD0-2167	no	RED
rml1p	L-Rhamnulose 1-phosphate	CPD-10791	no	RED
g3p	Glyceraldehyde 3-phosphate	Glycerol-1-phosphate	no	RED
adocbi	Adenosyl cobinamide	ADENOSYLCOBINAMIDE	yes	TTP
adocbip	Adenosyl cobinamide phosphate	ADENOSYLCOBINAMIDE-P	yes	TTP
adocbl	Adenosylcobalamin	ADENOSYLCOBALAMIN	yes	TTP
cbi	Cobinamide	COBINAMIDE	yes	TTP
cb1l	Cob(I)alamin	COB-I-ALAMIN	yes	TTP
sheme	Siroheme	SIROHEME	yes	TTP

<i>Composition</i>				
Abrv	Name (iJR904)	Retrieved	Comment	
actACP	Acetoacetyl-ACP	CPD-42	no	SNP

A.7.4 False negatives

<i>No structure annotated</i>			
Abrv	Name (iJR904)	Retrieved	Expected
ACP	acyl carrier protein	-	ACP
apoACP	apoprotein [ACP]	-	MYELIN-PROTEOLIPIDS
glutna	L-Glutamyl-tRNA(Glu)	-	CHARGED-GLT-TRNAS
glycogen	glycogen	-	GLYCOGENS
trdox	Oxidized thioredoxin	-	OX-THIOREDOXIN
trdrd	Reduced thioredoxin	-	RED-THIOREDOXIN
trnaglu	tRNA (Glu)	-	GLT-TRNAS
12dgr_EC	1,2-Diacylglycerol (E.coli) **	-	DIACYLGLYCEROL
pgp_EC	Phosphatidylglycerophosphate (Ecoli) **	-	L-1-PHOSPHATIDYL-GLYCEROL-P
ps_EC	phosphatidylserine (Ecoli) **	-	L-1-PHOSPHATIDYL-SERINE

<i>With structure annotated</i>			
Abrv	Name (iJR904)	Retrieved	Expected
3hmrsACP	R-3-hydroxy-myristoyl-ACP	-	R-3-HYDROXYMYRISTOYL-ACPS
acACP	Acetyl-ACP	-	ACETYL-ACP
ddcaACP	Dodecanoyl-ACP (n-C12:0ACP)	-	DODECANOYL-ACPS
hemeO	Heme O	-	HEME_O
malACP	Malonyl-[acyl-carrier protein]	-	MALONYL-ACP

A.7.5 One candidate

A.7.5.1 Identical false positives

- MCD: The retrieved candidate and expected entry appear to be duplicates in MetaCyc
- FAM: The retrieved candidate and expected entry are a parent and child ontology class
- ANO: The retrieved candidate and expected entry is the α or β anomer
- TTM: The retrieved candidate and expected entry are tautomers
- NAM: The *ijR904* has an ambiguous name and an incorrect representation assigned

Abrv	Name (<i>ijR904</i>)	Retrieved	Expected	Reason
5caiz	5-phosphoribosyl-5-carboxyaminoimidazole	CPD0-181	CPD-9005	MCD
kdo2lipid4p	KDO(2)-lipid IV(A) with palmitoleoyl	KDO2-PALMITOLEOYL- LIPID-IVA	CPD0-2265	MCD
fuc-L	L-Fucose	L-Fucopyranoses	L-FUCOSE	FAM
gal	D-Galactose	D-galactopyranose	D-GALACTOSE	FAM
glcur	D-Glucuronate	D-Glucopyranuronate	GLUCURONATE	FAM
man(e)	D-Mannose	D-mannopyranose	MANNOSE	FAM
lipidA	2,3-Bis(3-hydroxytetradecanoyl)-D-glucosaminyl-1,6-beta-D-2,3-bis(3-hydroxytetradecanoyl)-beta-D-glucosaminyl 1-phosphate	BISOHMYR-GLC	LIPID-IV-A	NAM
udcpp	Undecaprenyl phosphate	UNDECAPRENYL-P	CPD-9646	NAM
dhmp	Dihydroneopterin monophosphate	CPD-13344	DIHYDRONEOPTERIN-P	NAM
dtidp4d6dg	dTDP-4-dehydro-6-deoxy-D-glucose	DTDP-DEOH-DEOXY- GLUCOSE	DTDP-4-DEHYDRO-6- DEOXY-D-GALACTOSE	ANO*
f6p	D-Fructose 6-phosphate	CPD-15709	FRUCTOSE-6P	ANO
udpg	UDPglucose	UDP-GLUCOSE	CPD-12575	ANO
2dhgln	2-Dehydro-L-gulonate	CPD-13059	-	**
2ombl	2-Octaprenyl-3-methyl-6-methoxy-1,4-benzoquinol	OCTAPRENYL-METHYL- METHOXY-BENZQ	-	
5mtr	5-Methylthio-D-ribose	CPD-560	-	
6hmhpt	6-hydroxymethyl dihydropterin	AMINO-OH- HYDROXYMETHYL- DIHYDROPTERIDINE	-	TTM
6hmhptpp	6-hydroxymethyl-dihydropterin pyrophosphate	DIHYDROPTERIN-CH2OH- PP	-	TTM
acg5p	N-Acetyl-L-glutamyl 5-phosphate	N-ACETYL-GLUTAMYL-P		
adpglc	ADPglucose	ADP-D-GLUCOSE	-	*
gbbtn	gamma-butyrobetaine	GAMMA-BUTYROBETAINE	-	
malt	Maltose	MALTOSE	-	
pheme	Protoheme	PROTOHEME	-	
r5p	alpha-D-Ribose 5-phosphate	CPD-15318	-	
ssaltpp	Succinate semialdehyde-thiamin diphosphate anion	CPD0-2102	-	
tagdp-D	D-Tagatose 1,6-bisphosphate	TAGATOSE-1-6- DIPHOSPHATE	-	

* the structure in *ijR904* was mistakenly annotated with the α anomer.

** incorrect structure depiction in MNXRef.

A.7.5.2 Stereoisomer false positives

Abrv	Name (<i>ijR904</i>)	Retrieved	Expected	Comment
dttdp4addg	dTDP-4-amino-4,6-dideoxy-D-glucose	CPD-472	CPD-14020	MNXref mapped to galactose
2mcit	2-Methylcitrate	CPD-622	-	
hkndd	2-Hydroxy-6-oxonona-2,4-diene-1,9-dioate	CPD-157	-	
hkntd	2-hydroxy-6-ketononatrienedioate	CPD0-2184	-	
2dr1p	2-Deoxy-D-ribose 1-phosphate	DEOXY-D-RIBOSE-1-PHOSPHATE	-	
adphep-D,D	ADP-D-glycero-D-manno-heptose	ADP-D-GLYCERO-D-MANNO-HEPTOSE	-	
adphep-L,D	ADP-L-glycero-D-manno-heptose	ADP-L-GLYCERO-D-MANNO-HEPTOSE	-	
f1p	D-Fructose 1-phosphate	FRU1P	-	
fdp	D-Fructose 1,6-bisphosphate	FRUCTOSE-1,6-DIPHOSPHATE	-	

A.7.5.3 Structurally related false positives

- SNP: Correct structure not provided in MetaCyc
- RED: Retrieval deficiency, correct structure had missing or wrong stereo

Abrv	Name (<i>ijR904</i>)	Retrieved	Expected	Reason
actACP	Acetoacetyl-ACP	CPD-42	ACETOACETYL-ACPS	SNP
dhptd	4,5-dihydroxy-2,3-pentanedione	CPD0-2167	DIHYDROXYPENTANEDIONE	RED
rml1p	L-Rhamnulose 1-phosphate	CPD-10791	RHAMNULOSE-1P	RED
g3p	Glyceraldehyde 3-phosphate	Glycerol-1-phosphate	-	RED
ttdcea	tetradecenoate (n-C14:1)	CPD-7836	-	SNP

A.7.6 Two or more candidates

A.7.6.1 Two candidates, false positives

- FAM: Retrieved and expected candidate are family members (e.g. compound class)
- UFA: Unsaturated fatty acid with unspecified double bond location
- RED: Retrieval deficiency, correct structure had missing or wrong stereo

Abrv	Name (<i>ijR904</i>)	Retrieved	Expected	Reason
arab-L	L-Arabinose	CPD-15699	L-ARABINOSE	FAM
arab-L	L-Arabinose	L-arabinopyranose	L-ARABINOSE	FAM
fru	D-Fructose	CPD-15382	FRU	FAM
fru	D-Fructose	Fructofuranose	FRU	FAM
galur	D-Galacturonate	D-Galactopyranuronate	D-GALACTURONATE	FAM
galur	D-Galacturonate	CPD-15633	D-GALACTURONATE	FAM
glc-D	D-Glucose	Glucopyranose	D-GLUCOSE	FAM
glc-D	D-Glucose	CPD-15374	D-GLUCOSE	FAM
orn	Ornithine	CPD-217	25-DIAMINOPENTANOATE	FAM
orn	Ornithine	L-ORNITHINE	25-DIAMINOPENTANOATE	FAM
rib-D	D-Ribose	CPD-15818	CPD-10330, CPD-12043, CPD-12044, CPD-6001, PENTOSE-RING, RIBOSE	FAM
rib-D	D-Ribose	D-Ribofuranose	CPD-10330, CPD-12043, CPD-12044, CPD-6001, PENTOSE-RING, RIBOSE	FAM
rmn	L-Rhamnose	CPD-15405	L-RHAMNOSE	FAM
rmn	L-Rhamnose	L-rhamnopyranose	L-RHAMNOSE	FAM
xyl-D	D-Xylose	D-Xylopyranose	D-XYLOSE	FAM
xyl-D	D-Xylose	CPD-15377	D-XYLOSE	FAM
hdcea	hexadecenoate (n-C16:1)	CPD-9767	CPD-9245	UFA
hdcea	hexadecenoate (n-C16:1)	CPD-9768	CPD-9245	UFA
btcoa	Butanoyl-CoA	CPD-226	BUTYRYL-COA	RED
btcoa	Butanoyl-CoA	CROTONYL-COA	BUTYRYL-COA	RED
man6p	D-Mannose 6-phosphate	MANNOSE-6P	-	
man6p	D-Mannose 6-phosphate	CPD-15711	-	

A.7.6.2 More than two candidates, false positives

Abrv	Name (<i>ijR904</i>)	Retrieved	Expected
icit	Isocitrate	CPD-10748	-
icit	Isocitrate	CPD-10747	-
icit	Isocitrate	CPD-10745	-
icit	Isocitrate	THREO-DS-ISO-CITRATE	-
sbzcoa	O-Succinylbenzoyl-CoA	BENZOYLSUCCINYL-COA	-
sbzcoa	O-Succinylbenzoyl-CoA	SINAPOYL-COA	-
sbzcoa	O-Succinylbenzoyl-CoA	3-HYDROXY-3-4-METHYLPENT-3-EN-1-YLG-COA	-
sbzcoa	O-Succinylbenzoyl-CoA	CARBOXYMETHYL-HYDROXYPHENYLPROPCOA	-
sbzcoa	O-Succinylbenzoyl-CoA	CPD-6972	-
ocdcea	octadecenoate (n-C18:1)	STEARIC_ACID	-
ocdcea	octadecenoate (n-C18:1)	OLEATE-CPD	-
ocdcea	octadecenoate (n-C18:1)	CPD-8477	-
ocdcea	octadecenoate (n-C18:1)	CPD-14260	-
ocdcea	octadecenoate (n-C18:1)	CPD-9247	-
ocdcea	octadecenoate (n-C18:1)	CREPENYNATE	-
ocdcea	octadecenoate (n-C18:1)	CPD-8203	-
ocdcea	octadecenoate (n-C18:1)	CPD-8478	-
ocdcea	octadecenoate (n-C18:1)	LINOLEIC_ACID	-
ocdcea	octadecenoate (n-C18:1)	9-CIS11-TRANS-OCTADECADIENOATE	-
ocdcea	octadecenoate (n-C18:1)	CPD-15704	-
ocdcea	octadecenoate (n-C18:1)	LINOLENIC_ACID	-
ocdcea	octadecenoate (n-C18:1)	CPD-10245	-
ocdcea	octadecenoate (n-C18:1)	CPD-12652	-
ocdcea	octadecenoate (n-C18:1)	CPD-8229	-
ocdcea	octadecenoate (n-C18:1)	CPD-8228	-
ocdcea	octadecenoate (n-C18:1)	CPD-8231	-
ocdcea	octadecenoate (n-C18:1)	CPD-8117	-
ocdcea	octadecenoate (n-C18:1)	CPD-14869	-
ocdcea	octadecenoate (n-C18:1)	CPD-12654	-
ocdcea	octadecenoate (n-C18:1)	CPD-12653	-
ocdcea	octadecenoate (n-C18:1)	CPD-14868	-

A.7.6.3 Two candidates, true positives

Abvr	Name (<i>ijR904</i>)	Retrieved	Expected
alltn	Allantoin	S-ALLANTOIN	ALLANTOIN, S-ALLANTOIN
alltn	Allantoin	R-ALLANTOIN	ALLANTOIN, S-ALLANTOIN
crncoa	CarnitinyI-CoA	D-CARNITINYI-COA	L-CARNITINYI-COA
crncoa	CarnitinyI-CoA	L-CARNITINYI-COA	L-CARNITINYI-COA
cyan	Cyanide	CPD-13584	CPD-13584, HCN
cyan	Cyanide	HCN	CPD-13584, HCN
fruur	D-Fructuronate	CPD-12536	CPD-12536, CPD-12537, FRUCTURONATE
fruur	D-Fructuronate	CPD-12537	CPD-12536, CPD-12537, FRUCTURONATE
g1p	D-Glucose 1-phosphate	CPD-448	GLC-1-P
g1p	D-Glucose 1-phosphate	GLC-1-P	GLC-1-P
D-Glucose 6-phosphate	D-glucose-6-phosphate	D-GLUCOSE-6-PHOSPHATE	
g6p	D-Glucose 6-phosphate	CPD-9755	D-GLUCOSE-6-PHOSPHATE
gmhep17bp	D-Glycero-D-manno-heptose 1,7-bisphosphate	D-BETA-D-HEPTOSE-17- DIPHOSPHATE	D-BETA-D-HEPTOSE-17- DIPHOSPHATE
gmhep17bp	D-Glycero-D-manno-heptose 1,7-bisphosphate	CPD-12334	D-BETA-D-HEPTOSE-17- DIPHOSPHATE
gmhep1p	D-Glycero-D-manno-heptose 1-phosphate	CPD-12335	D-BETA-D-HEPTOSE-1-P
gmhep1p	D-Glycero-D-manno-heptose 1-phosphate	D-BETA-D-HEPTOSE-1-P	D-BETA-D-HEPTOSE-1-P
nh4	ammonium	AMMONIA	AMMONIA, AMMONIUM
nh4	ammonium	AMMONIUM	AMMONIA, AMMONIUM
shcl	Sirohydrochlorin	SIROHYDROCHLORIN	DIHYDROSIROHYDROCHLORIN
shcl	Sirohydrochlorin	DIHYDROSIROHYDROCHLORIN	DIHYDROSIROHYDROCHLORIN
so4	Sulfate	SULFATE	HSO4, SULFATE
so4	Sulfate	HSO4	HSO4, SULFATE
tre	Trehalose	TREHALOSE	ALPHABETA-TREHALOSE, TREHALOSE
tre	Trehalose	ALPHABETA-TREHALOSE	ALPHABETA-TREHALOSE, TREHALOSE
uaagmda	Undecaprenyl-diphospho-N- acetylmutaromoyl-(N- acetylglucosamine)-L-ala-D-glu-meso- 2,6-diaminopimeloyl-D-ala-D-ala	C6	C6
uaagmda	Undecaprenyl-diphospho-N- acetylmutaromoyl-(N- acetylglucosamine)-L-ala-D-glu-meso- 2,6-diaminopimeloyl-D-ala-D-ala	Peptidoglycans	C6

A.7.6.4 More than two candidates, true positives

Abrv	Name (<i>iJR904</i>)	Retrieved	Expected	
fe2	Fe ²⁺	FE+3	FE+2	oxidation
fe2	Fe ²⁺	FE+4	FE+2	oxidation
fe2	Fe ²⁺	FE+2	FE+2	oxidation
gmhep7p	D-Glycero-D-manno-heptose 7-phosphate	CPD-12542	D-ALPHABETA-D-HEPTOSE- 7-PHOSPHATE	
gmhep7p	D-Glycero-D-manno-heptose 7-phosphate	CPD-12543	D-ALPHABETA-D-HEPTOSE- 7-PHOSPHATE	
gmhep7p	D-Glycero-D-manno-heptose 7-phosphate	D-ALPHABETA-D-HEPTOSE- 7-PHOSPHATE	D-ALPHABETA-D-HEPTOSE- 7-PHOSPHATE	
h2o	H ₂ O	OH	CPD-12377, OH, WATER	
h2o	H ₂ O	WATER	CPD-12377, OH, WATER	
h2o	H ₂ O	OXONIUM	CPD-12377, OH, WATER	
h2s	Hydrogen sulfide	HS	CPD-7046, HS	
h2s	Hydrogen sulfide	CPD-7046	CPD-7046, HS	
h2s	Hydrogen sulfide	CPD-846	CPD-7046, HS	
hco3	Bicarbonate	HCO ₃	CO ₃ , H ₂ CO ₃ , HCO ₃	
hco3	Bicarbonate	CO ₃	CO ₃ , H ₂ CO ₃ , HCO ₃	
hco3	Bicarbonate	H ₂ CO ₃	CO ₃ , H ₂ CO ₃ , HCO ₃	
so3	Sulfite	H ₂ SO ₃	H ₂ SO ₃ , HSO ₃ , SO ₃	
so3	Sulfite	HSO ₃	H ₂ SO ₃ , HSO ₃ , SO ₃	
so3	Sulfite	SO ₃	H ₂ SO ₃ , HSO ₃ , SO ₃	

A.8 Alignment of *iYO844* and *iBsu1103*

A.8.1 Without structure annotations

<i>Polysaccharides / Mixtures</i>	
Abbreviation	Name
14bxyl	1,4-beta-D-Xylan
14glun	1,4-alpha-D-Glucan
Larab	alpha-L-Arabinan
dextrin	Dextrin
glycogen	glycogen
galman	Galactomannan
galogs	Galactose oligosaccharide
starch	Starch
bilea	Bile acid
<i>Proteins / macromolecules</i>	
Abbreviation	Name

ACP	acyl carrier protein
apoACP	apoprotein [acyl carrier protein]
trdox	Oxidized thioredoxin
trdrd	Reduced thioredoxin

Transfer RNA

Abbreviation	Name
fmettrna	N-Formylmethionyl-tRNA
glutrna	L-Glutamyl-tRNA(Glu)
mettrna	L-Methionyl-tRNA (Met)
trnaglu	tRNA (Glu)
trnamet	tRNA(Met)

Virtual metabolites

Abbreviation	Name
gtca1-45_BS	glycerol teichoic acid (n=45), unlinked, unsubstituted
teich-45_BS	teichuronic acid (GlcA + GalNAc, 45 repeating unit)
unk2_BS	unknown 2
12dag3p_BS	1,2-diacyl-sn-glycerol 3-phosphate
12dgr_BS	1,2-diacylglycerol
1ag3p_BS	1-Acyl-sn-glycerol 3-phosphate
cdlp_BS	cardiolipin (B. subtilis)
cdpdag_BS	CDPdiacylglycerol (B. subtilis)
d12dg_BS	diglucosyl-1,2 diacylglycerol
gtca2-45_BS	glycerol teichoic acid (n=45), unlinked, D-ala substituted
gtca3-45_BS	glycerol teichoic acid (n=45), unlinked, glucose substituted
lipo1-24_BS	lipoteichoic acid (n=24), linked, glucose substituted
lipo2-24_BS	lipoteichoic acid (n=24), linked, N-acetyl-D-glucosamine
lipo3-24_BS	lipoteichoic acid (n=24), linked, D-alanine substituted
lipo4-24_BS	lipoteichoic acid (n=24), linked, unsubstituted
lysylpgly_BS	lysylphosphatidylglycerol
m12dg_BS	monoglucosyl-1,2 diacylglycerol
peptido_BS	Peptidoglycan subunit of Bacillus subtilis
pgly_BS	phosphatidylglycerol (B. subtilis)
pglyp_BS	phosphatidylglycerophosphate (B. subtilis)
ps_BS	phosphatidylserine (B. subtilis)
psetha_BS	phosphatidylethanolamine (B. subtilis)
t12dg_BS	triglucosyl-1,2 diacylglycerol
tcam_BS	minor teichoic acid (acetylglactosamine glucose phosphate, n=30)

Missed annotation

Abbreviation	Name
uagmda	Undecaprenyl-diphospho-N-acetylmuramoyl-L-alanyl-D-glutamyl-meso-2,6-diaminopimeloyl-D-alanyl-D-alanine

A.8.2 False negatives

No structure annotated

Abrv	Name (<i>iYO844</i>)	Retrieved	Name (<i>iBsu1103</i>)	Expected	Reason
14glun	1,4-alpha-D-Glucan	-	-	cpd11735	
ACP	acyl carrier protein	-	-	cpd11493	
Larab	alpha-L-Arabinan	-	-	cpd12115	
apoACP	apoprotein [acyl carrier protein]	-	-	cpd12370	
dextrin	Dextrin	-	-	cpd11594	
glutrna	L-Glutamyl-tRNA(Glu)	-	-	cpd12227	
glycogen	glycogen	-	-	cpd00155	
teich-45_BS	teichuronic acid (GlcA + GalNAc, 45 repeating unit)	-	-	cpd15634	
trdox	Oxidized thioredoxin	-	-	cpd11420	
trdrd	Reduced thioredoxin	-	-	cpd11421	
trnaglu	tRNA (Glu)	-	-	cpd11912	
uagmda	Undecaprenyl-diphospho-N-acetylmuramoyl-L-alanyl-D-glutamyl-meso-2,6-diaminopimeloyl-D-alanyl-D-alanine	-	-	cpd03494	
tcam_BS	minor teichoic acid (acetylglactosamine glucose phosphate, n=30)	-	-	cpd11459	

With structure annotated

Abrv	Name (<i>iYO844</i>)	Retrieved	Name (<i>iBsu1103</i>)	Expected	Reason
antim	Antimonite	-	-	cpd11598	salt, hydrated
ficytc	Ferricytochrome c	-	-	cpd00109	tetrapyrrole
focytc	Ferrocycytochrome c	-	-	cpd00110	tetrapyrrole
hemeC	Heme C	-	-	cpd14553	tetrapyrrole
hemeD	Heme D	-	-	cpd15607	tetrapyrrole

A.8.3 One candidate

A.8.3.1 Identical false positives

- DUP: Duplicate in *iBsu1103*
- WSA: Wrong structure annotation in *iBsu1103*

Identical

Abrv	Name (<i>iYO844</i>)	Retrieved	Name (<i>iBsu1103</i>)	expected	Reason
gbbtn	gamma-butyrobetaine	cpd00870	4-Trimethylammoniobutanoate	cpd15471	DUP
hpdca	heptadecanoate (C17:0)	cpd11431	fa11	-	WSA
ptdca	pentadecanoate (C15:0)	cpd11436	fa3	-	WSA

A.8.3.2 Stereoisomer false positives

- UFA: Unsaturated fatty acid with unspecified double bond location
- RDP: Reverse dipeptide *iBsu1103*
- NCM: New candidate match
- BCE: Better candidate exists, another *iYO844* matched the candidate as identical
- MPS: Misrepresented polysaccharide in *iBsu1103*

Diastereomers

Abrv	Name (<i>iYO844</i>)	retrieved	Name (<i>iBsu1103</i>)	expected	Reason
diala-L	Ala-Ala	cpd00731	D-Alanyl-D-alanine	-	BCE
inospp1	1D-myo-inositol	cpd02780	D-myo-Inositol	-	NCM
	1,3,4,5,6-pentakisphosphate		1,2,4,5,6-pentakisphosphate		
udcpdp	Undecaprenyl diphosphate	cpd02229	Undecaprenyl diphosphate	-	BCE

Unspecified

Abrv	Name (<i>iYO844</i>)	retrieved	Name (<i>iBsu1103</i>)	expected	Reason
hdcoa	Hexadecenoyl-CoA (n-C16:1CoA)	cpd03126	trans-Hexadec-2-enoyl-CoA	cpd15238	UFA

More specific

Abrv	Name (<i>iYO844</i>)	retrieved	Name (<i>iBsu1103</i>)	expected	Reason
ala-L-met-L	Ala-Met	cpd11590	met-L-ala-L	-	RDP
fa12coa	Anteiso-C17:0 CoA	cpd11434	fa12coa	-	NCM
fa1coa	Iso-C14:0 CoA	cpd11435	fa1coa	-	NCM
fa3coa	Iso-C15:0 CoA	cpd11437	fa3coa	-	NCM
fa4coa	Anteiso-C15:0 CoA	cpd11439	fa4coa	-	NCM
mmalsa	(S)-Methylmalonate semialdehyde	cpd00287	2-Methyl-3-oxopropanoate	-	BCE

Less specific (single)

Abrv	Name (<i>iYO844</i>)	Retrieved	Name (<i>iBsu1103</i>)	Expected	Reason
malthx	Maltohexaose	cpd11594	Dextrin	-	MPS
maltpt	Maltopentaose	cpd11735	Amylose	-	MPS

A.8.3.3 Structurally related false positives

- RED: Retrieval deficiency, correct structure had missing or wrong stereo

<i>Skeleton</i>					
Abrv	Name (<i>iYO844</i>)	Retrieved	Name (<i>iBsu1103</i>)	Expected	Reason
2hbut	2-Hydroxybutyrate	cpd00094	2-Oxobutanoate	-	
fa10	Fatty acid (Anteiso-C17:1)	cpd11433	fa12	-	
fa10coa	Anteiso-C17:1 CoA	cpd11434	fa12coa	-	
fa5coa	Iso-C16:1 CoA	cpd11441	fa6coa	-	
fa9coa	Iso-C17:1 CoA	cpd11432	fa11coa	-	
fadh2	FADH2	cpd00015	FAD	cpd00982	RED
glcurn	D-Glucurone	cpd00765	L-Gulono-1,4-lactone	-	
mlthf	5,10-Methylenetetrahydrofolate	cpd00347	5,10-Methenyltetrahydrofolate	cpd00125	RED

<i>Composition</i>					
Abrv	Name (<i>iYO844</i>)	Retrieved	Name (<i>iBsu1103</i>)	Expected	Reason
6ampenc	6-Aminopenicillanate	cpd11590	met-L-ala-L	-	
L-thrgly	Thr-Gly	cpd00344	N-Formimino-L-glutamate	-	
ala-L-asn-L	Ala-Asn	cpd11580	Gly-Gln	-	
ala-L-ile-L	Ala-Ile	cpd11583	Ala-Leu	-	
ala-L-ser-L	Ala-Ser	cpd00344	N-Formimino-L-glutamate	-	
gly-cys-L	Gly-Cys	cpd01017	Cys-Gly	-	
gly-tyr-L	Gly-Try	cpd00689	Porphobilinogen	-	
glygly	Glycylglycine	cpd00132	L-Asparagine	-	
octa	octanoate	cpd00430	Phenyl acetate	-	
octal	octanal	cpd00464	Phenylacetaldehyde	-	
pydx5p	Pyridoxal 5'-phosphate	cpd00507	sn-glycero-3-Phosphocholine	-	
ser-L-ala-L	Ser-Ala	cpd00344	N-Formimino-L-glutamate	-	
ser-L-ser-L	Ser-Ser	cpd11589	gly-asp-L	-	
thr-L-leu-L	Thr-Leu	cpd00689	Porphobilinogen	-	

A.8.3.4 Structurally related true positives

- TTP: The structures are tetrapyrroles and have differences in representation
- RDP: Reverse dipeptide *iBsu1103*
- NOX: Unstandardised nitrogen oxide
- UPL: Unspecified protonation location
- COA: Coenzyme A attachment
- GAA: Gamma attached amino acid

<i>Skeleton</i>				
Abrv	Name (<i>iYO844</i>)	Retrieved	Name (<i>iBsu1103</i>)	Reason
cb12	Cob(II)alamin	cpd00423	Cob(II)alamin	TTP
ferrich	Ferrichrome	cpd03724	Ferrichrome	TTP
hemeA	Heme A	cpd11312	Heme A	TTP
hemeO	Heme O	cpd11313	Heme O	TTP
no3	Nitrate	cpd00209	Nitrate	NOX
pheme	Protoheme	cpd00028	Heme	TTP

<i>Composition</i>				
Abrv	Name (<i>iYO844</i>)	Retrieved	Name (<i>iBsu1103</i>)	Reason
ala-L-gln-L	Ala-Gln	cpd11587	Ala-Gln	RDP
ala-L-leu-L	Ala-Leu	cpd11583	Ala-Leu	RDP
met-L-ala-L	met-L-ala-L	cpd11590	met-L-ala-L	RDP
icit	Isocitrate	cpd00260	Isocitrate	UPL
5d4dglcr	5-Dehydro-4-deoxy-D-glucarate	cpd00515	5-Dehydro-4-deoxy-D-glucarate	UPL
quln	Quinolate	cpd02333	Pyridine-2,3-dicarboxylate	UPL
sbzcoa	O-Succinylbenzoyl-CoA	cpd02021	2-Succinylbenzoyl-CoA	COA
uaagmda	Undecaprenyl-diphospho-N-acetylmuramoyl-(N-acetylglucosamine)-L-alanyl-D-glutamyl-meso-2,6-diaminopimeloyl-D-alanyl-D-alanine	cpd03495	Undecaprenyl-diphospho-N-acetyluramoyl-(N-acetylglucosamine)-L-alanyl-D-glutamyl-meso-2,6-diaminopimeloyl-D-alanyl-D-alanine	GAA
sheme	Siroheme	cpd00557	Siroheme	TTP
uppg3	Uroporphyrinogen III	cpd00774	Uroporphyrinogen III	TTP

A.8.4 Two or more candidates

A.8.4.1 Two candidates, false positives

The crotonobetaine entry in *iBsu1103* had a different structure representation.

<i>Skeleton</i>				
Abrv	Name (<i>iYO844</i>)	Retrieved	Name (<i>iBsu1103</i>)	Expected
ctbt	crotonobetaine	cpd00870	4-Trimethylammoniobutanoate	cpd08305
ctbt	crotonobetaine	cpd15471	gamma-butyrobetaine	cpd08305
6pthp	6-Pyruvoyl-5,6,7,8-tetrahydropterin	cpd00233	Tetrahydrobiopterin	-
6pthp	6-Pyruvoyl-5,6,7,8-tetrahydropterin	cpd00231	Dihydrobiopterin	-

<i>Composition</i>				
Abrv	Name (<i>iYO844</i>)	Retrieved	Name (<i>iBsu1103</i>)	Expected
L-sergly	L-Serylglycine	cpd00247	Orotate	-
L-sergly	L-Serylglycine	cpd00282	(S)-Dihydroorotate	-
dca	Decanoate	cpd03712	1,2-Dihydronaphthalene-1,2-diol	-
dca	Decanoate	cpd01932	Naphthalene-1,2-diol	-
dcal	decanal	cpd10483	(1R,2S)-Naphthalene 1,2-oxide	-
dcal	decanal	cpd10484	(1S,2R)-Naphthalene 1,2-oxide	-
fa11coa	Iso-C17:0 CoA	cpd11432	fa11coa	-
fa11coa	Iso-C17:0 CoA	cpd11434	fa12coa	-
fa9	Fatty acid (Iso-C17:1)	cpd11431	fa11	-
fa9	Fatty acid (Iso-C17:1)	cpd11433	fa12	-
hxa	Hexanoate	cpd00077	Catechol	-
hxa	Hexanoate	cpd00415	p-Benzenediol	-
maltttr	Maltotetraose	cpd01133	Stachyose	-
maltttr	Maltotetraose	cpd15302	glycogen(n-1)	-
mmal	Methylmalonate	cpd00106	Fumarate	-
mmal	Methylmalonate	cpd00036	Succinate	-
thr-L-gln-L	Thr-Gln	cpd03407	gamma-Glutamyl-beta-cyanoalanine	-
thr-L-gln-L	Thr-Gln	cpd00367	Cytidine	-

A.8.4.2 More than two candidates, false positives

The 4 entries matched by H⁺ were not loaded correctly and obtained the same hash code 'nil' as the proton.

<i>Skeleton</i>				
Abrv	Name (<i>iYO844</i>)	Retrieved	Name (<i>iBsu1103</i>)	Expected
Lglyald	L-Glyceraldehyde	cpd00100	Glycerol	-
Lglyald	L-Glyceraldehyde	cpd00448	D-Glyceraldehyde	-
Lglyald	L-Glyceraldehyde	cpd00157	Glycerone	-
<i>Composition</i>				
Abrv	Name (<i>iYO844</i>)	Retrieved	Name (<i>iBsu1103</i>)	Expected
fa5	Fatty acid (Iso-C16:1)	cpd00214	Hexadecanoic acid	-
fa5	Fatty acid (Iso-C16:1)	cpd15237	hexadecenoate	-
fa5	Fatty acid (Iso-C16:1)	cpd11440	fa6	-
sarcs	Sarcosine	cpd00035	L-Alanine	-
sarcs	Sarcosine	cpd00117	D-Alanine	-
sarcs	Sarcosine	cpd00085	beta-Alanine	-
thr-L-thr-L	Thr-Thr	cpd11586	ala-L-glu-L	-
thr-L-thr-L	Thr-Thr	cpd15388	L-alanine-L-glutamate	-
thr-L-thr-L	Thr-Thr	cpd15385	L-alanine-D-glutamate	-
fa6coa	Iso-C16:0 CoA	cpd00134	Palmitoyl-CoA	-
fa6coa	Iso-C16:0 CoA	cpd15238	Hexadecenoyl-CoA	-
fa6coa	Iso-C16:0 CoA	cpd11441	fa6coa	-
fa6coa	Iso-C16:0 CoA	cpd03126	trans-Hexadec-2-enoyl-CoA	-
g16bp	D-Glucose 1,6-bisphosphate	cpd00290	D-Fructose 1,6-bisphosphate	-
g16bp	D-Glucose 1,6-bisphosphate	cpd02371	D-Tagatose 1,6-bisphosphate	-
g16bp	D-Glucose 1,6-bisphosphate	cpd00290	D-Fructose 1,6-bisphosphate	-
g16bp	D-Glucose 1,6-bisphosphate	cpd02371	D-Tagatose 1,6-bisphosphate	-
h	H+	cpd15662	45(Glucosyl-phosphoglyceryl)-N-Acetyl-beta-D-mannosaminy-1,4-N-acetyl-D-glucosaminyldiphosphoundecaprenol	cpd00067
h	H+	cpd15663	45(Alanyl-phosphoglyceryl)-N-Acetyl-beta-D-mannosaminy-1,4-N-acetyl-D-glucosaminyldiphosphoundecaprenol	cpd00067
h	H+	cpd15668	glycerol teichoic acid (n=45), linked, D-ala substituted	cpd00067
h	H+	cpd15669	glycerol teichoic acid (n=45), linked, glucose substituted	cpd00067
hpa	heptanoate	cpd04117	4-Methylcatechol	-
hpa	heptanoate	cpd01871	2,3-Dihydroxytoluene	-
hpa	heptanoate	cpd01553	Salicyl alcohol	-
hpa	heptanoate	cpd00153	Benzoate	-

A.8.4.3 Two candidates, true positives

- DUP: Duplicate entry in *iBsu1103*
- OXD: Different oxidation states
- USC: Unrecognised stereochemistry (Fischer projection)
- USP: Unspecified stereochemistry

- WSA: Wrong structure annotation in *iBsu1103*
- TTP: The structures are tetrapyrroles and have differences in representation

Identical

Abrv	Name (<i>iYO844</i>)	Retrieved	Name (<i>iBsu1103</i>)	Expected	Reason
fe2	Fe2+	cpd10515	Fe2+	cpd10515	OXD
fe2	Fe2+	cpd10516	Fe3+	cpd10515	OXD
fe3	Fe3+	cpd10515	Fe2+	cpd10516	OXD
fe3	Fe3+	cpd10516	Fe3+	cpd10516	OXD
hdca	hexadecanoate (n-C16:0)	cpd00214	Hexadecanoic acid	cpd00214	DUP
hdca	hexadecanoate (n-C16:0)	cpd11440	fa6	cpd00214	DUP
ttzca	tetradecanoate (C14:0)	cpd11430	fa1	cpd03847	DUP
ttzca	tetradecanoate (C14:0)	cpd03847	Tetradecanoic acid	cpd03847	DUP

Stereoisomer

Abrv	Name (<i>iYO844</i>)	Retrieved	Name (<i>iBsu1103</i>)	Expected	Reason
galctr-D	D-Galactarate	cpd00652	D-Galactarate	cpd00652	USC
galctr-D	D-Galactarate	cpd00609	D-Glucarate	cpd00652	USC
gln-D	D-Gluconate	cpd00403	D-Mannonate	cpd00222	USC
gln-D	D-Gluconate	cpd00222	D-Gluconic acid	cpd00222	USC
glcr	D-Glucarate	cpd00652	D-Galactarate	cpd00609	USC
glcr	D-Glucarate	cpd00609	D-Glucarate	cpd00609	USC
madg	alpha-Methyl-D-glucoside	cpd15584	alpha-Methyl-D-glucoside	cpd15584	USC
madg	alpha-Methyl-D-glucoside	cpd15585	beta-Methylglucoside	cpd15584	USC
mana	D-Mannonate	cpd00403	D-Mannonate	cpd00403	USC
mana	D-Mannonate	cpd00222	D-Gluconic acid	cpd00403	USC
mbdg	beta-Methylglucoside	cpd15584	alpha-Methyl-D-glucoside	cpd15585	USC
mbdg	beta-Methylglucoside	cpd15585	beta-Methylglucoside	cpd15585	USC

Composition

Abrv	Name (<i>iYO844</i>)	Retrieved	Name (<i>iBsu1103</i>)	Expected	Reason
fa1	Fatty acid (Iso-C14:0)	cpd11430	fa1	cpd11430	WSA
fa1	Fatty acid (Iso-C14:0)	cpd03847	Tetradecanoic acid	cpd11430	WSA
fa11	Fatty acid (Iso-C17:0)	cpd11431	fa11	cpd11431	WSA
fa11	Fatty acid (Iso-C17:0)	cpd11433	fa12	cpd11431	WSA
fa3	Fatty acid (Iso-C15:0)	cpd11438	fa4	cpd11436	WSA
fa3	Fatty acid (Iso-C15:0)	cpd11436	fa3	cpd11436	WSA
shcl	dihydrosirohydrochlorin	cpd03426	Sirohydrochlorin	cpd01620	TTP
shcl	dihydrosirohydrochlorin	cpd01620	Precorin 2	cpd01620	TTP
srch	sirohydrochlorin	cpd03426	Sirohydrochlorin	cpd03426	TTP
srch	sirohydrochlorin	cpd01620	Precorin 2	cpd03426	TTP
ggdp	Geranylgeranyl diphosphate	cpd08211	trans,trans,cis-Geranylgeranyl diphosphate	cpd00289	USP
ggdp	Geranylgeranyl diphosphate	cpd00289	Geranylgeranyl diphosphate	cpd00289	USP

A.8.4.4 More than two candidates, true positives

- RDP: Reverse dipeptide *iBsu1103*
- USC: Unrecognised stereochemistry (Fischer projection)
- UFA: Unsaturated fatty acid with unspecified double bond location
- WSA: Wrong structure annotation in *iBsu1103*

<i>Stereoisomer</i>					
Abrv	Name (<i>iYO844</i>)	Retrieved	Name (<i>iBsu1103</i>)	Expected	Reason
ala-L-glu-L	ala-L-glu-L	cpd11586	ala-L-glu-L	cpd11586	RDP
ala-L-glu-L	ala-L-glu-L	cpd15388	L-alanine-L-glutamate	cpd11586	RDP
ala-L-glu-L	ala-L-glu-L	cpd15385	L-alanine-D-glutamate	cpd11586	RDP
galt	Galactitol	cpd00588	D-Sorbitol	cpd01171	USC
galt	Galactitol	cpd00314	Mannitol	cpd01171	USC
galt	Galactitol	cpd01171	Galactitol	cpd01171	USC
galt	Galactitol	cpd01187	L-Glucitol	cpd01171	USC
mn1	D-Mannitol	cpd00588	D-Sorbitol	cpd00314	USC
mn1	D-Mannitol	cpd00314	Mannitol	cpd00314	USC
mn1	D-Mannitol	cpd01171	Galactitol	cpd00314	USC
mn1	D-Mannitol	cpd01187	L-Glucitol	cpd00314	USC
sbt-D	D-Sorbitol	cpd00588	D-Sorbitol	cpd00588	USC
sbt-D	D-Sorbitol	cpd00314	Mannitol	cpd00588	USC
sbt-D	D-Sorbitol	cpd01171	Galactitol	cpd00588	USC
sbt-D	D-Sorbitol	cpd01187	L-Glucitol	cpd00588	USC

<i>Skeleton</i>					
Abrv	Name (<i>iYO844</i>)	Retrieved	Name (<i>iBsu1103</i>)	Expected	Reason
hdcea	hexadecenoate (n-C16:1)	cpd00214	Hexadecanoic acid	cpd15237	UFA
hdcea	hexadecenoate (n-C16:1)	cpd11440	fa6	cpd15237	UFA
hdcea	hexadecenoate (n-C16:1)	cpd15237	hexadecenoate	cpd15237	UFA

<i>Composition</i>					
Abrv	Name (<i>iYO844</i>)	Retrieved	Name (<i>iBsu1103</i>)	Expected	Reason
fa6	Fatty acid (iso-C16:0)	cpd00214	Hexadecanoic acid	cpd11440	WSA
fa6	Fatty acid (iso-C16:0)	cpd15237	hexadecenoate	cpd11440	WSA
fa6	Fatty acid (iso-C16:0)	cpd11440	fa6	cpd11440	WSA

Appendix B

Chapter 3 supplementary information

B.1 Incorrect MIRIAM annotations

The following annotations were found in <http://www.nature.com/ncomms/2014/140114/ncomms4083/full/ncomms4083.html#supplementary-information> by a Metingear user. Identifier URNs and URLs provide a means to encode an identifier an a resource. To be interrupted by other tools these must be encoded correctly. A developer from the MIRIAM registry (Nick Juty) has confirmed theses encodings are invalid.

Encoding multiple annotations requires separate resolvable identifiers.

```
<rdf:li rdf:resource="urn:miriam:ec-code:EC:1.1.1.62;EC:1.1.1.64"/>
```

is correctly encoded as

```
<rdf:li rdf:resource="urn:miriam:ec-code:EC:1.1.1.62"/>
```

```
<rdf:li rdf:resource="urn:miriam:ec-code:EC:1.1.1.64"/>
```

The TC classification numbers are not the same as EC and require a different resource description

```
<rdf:li rdf:resource="urn:miriam:ec-code:TCDB:2.A.29.8.3"/>
```

is correctly encoded as

```
<rdf:li rdf:resource="urn:miriam:tcd:2.A.29.8.3"/>
```

B.2 SBML species definitions

The following listing shows a selection of metabolite definitions encoded in SBML for different reconstructions. Although the SBML syntax is consistent, different conventions and styles are present.

```
<!-- Acinetobacter baumannii AYE -->
<species name="C180ACP" compartment="cell" initialAmount="1" boundaryCondition="false" />
<!-- Burkholderia cenocepacia J2315 -->
<species id="EC0127" name="(S)-Malate[e]" charge="-2" boundaryCondition="false"
  compartment="Extra_organism" />
<!-- Clostridium thermocellum ATCC 27405 -->
<species id="M_acorn_c" name="." compartment="Cytosol"/>
<!-- Mycoplasma genitalium -->
<species id="M_10fthf_c" name="M_10-Formyltetrahydrofolate_C20H21N7O7"
  compartment="Cytosol" charge="-2" boundaryCondition="false"/>
<!-- Mycobacterium tuberculosis -->
<species id="M_10fthf_c" name="10-Formyltetrahydrofolate" compartment="C_c" >
  <notes>
    <html xmlns="http://www.w3.org/1999/xhtml" ><p>FORMULA: C20H21N7O7</p></html>
  </notes>
</species>
<!-- Pseudomonas putida -->
<species id="T4HP" compartment="cell" initialAmount="1" boundaryCondition="false" />
<!-- Sinorhizobium meliloti -->
<species id="M_n2" name="nitrogen" compartment="Cytosol"/>
<!-- Rhodobacter sphaeroides -->
<species compartment="Cytosol" id="CPD0001" name="H2O" charge="0" boundaryCondition="false"/>
<!-- Vibrio vulnificus -->
<species id="C180ACP" compartment="cell" initialAmount="1" boundaryCondition="false" />
```

Table B.1: Examples of normalised metabolite names

Original	Normalised
hexa-decane	hexadecane
ATP(3-)	atp
bicyclo[2.2.2]octane	bicyclo222octane
<i>(D)</i>-Alanine	dalanine
α-D-glucose	adglucose

B.3 Normalisation of metabolite names

Differences in metabolite names may be purely stylistic or typographical. To reduce false negatives due to these differences, names are normalised before comparison. Standard adjustments are applied, including: converting to lower case and removing white space, HTML tags, accents, and nomenclature punctuation (e.g. brackets, hyphens, bullets) (Tab. B.1). Nomenclature punctuation is only removed for comparison and the original name remains unmodified. Greek characters encoded as HTML entities (α), Unicode characters (α), or spelt (alpha) are also standardised.

B.4 Ring closing SMARTS patterns

The following SMARTS queries attempt to match chain form carbohydrates:

[O!RH1]-[C!R!\$(C=O)]-[C!R]-[C!R]-[C!R]-[C!RH1]=O pyranose

[O!RH1]-[C!R!\$(C=O)]-[C!R]-[C!R]-[C!RH1]=O furanose

The pseudo code outlines the operations that form a pyranose ring from a bijection of the query SMARTS atoms.

```
m = q -> t          // bijection from query to target
SIGMA(m[0], m[5])    // new sigma bond between the first and sixth atom (pyranose)
SIGMA(m[5], m[6])    // convert pi bond to sigma bond
MOVE_H(m[0], m[6])   // move the hydrogen from first to seventh atom
```

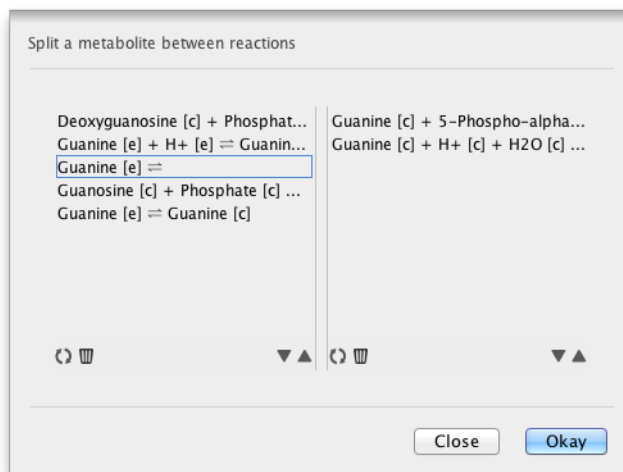



Figure B.1: A metabolite is split by selecting two sets of reactions. Each newly created metabolite will be involved in only one set.

B.5 Convenience functionality

Beyond the modification and annotation, Metingear provides several operations for exploring and extending metabolic reconstructions.

Split and merge metabolites Metabolite entities can be split and merged. A metabolite is split by selecting the reactions that each newly created metabolite will belong to (Fig. B.1). Metabolites cannot be split across a reaction (i.e. transporters) but this functionality can be achieved by editing the reaction after the initial split is performed. Multiple metabolites can also be merged into a single entity. Metabolites to merge may be selected manually or automatically using name, identifier or chemical structure.

Metabolic choke points A metabolic choke point is a reaction that either uniquely consumes or produces a metabolite^[251]. Discovered choke points are selected in the entity table, with a flag and note is added to each reaction. The method is based on connectivity and does not determine if the reaction is active through simulation.

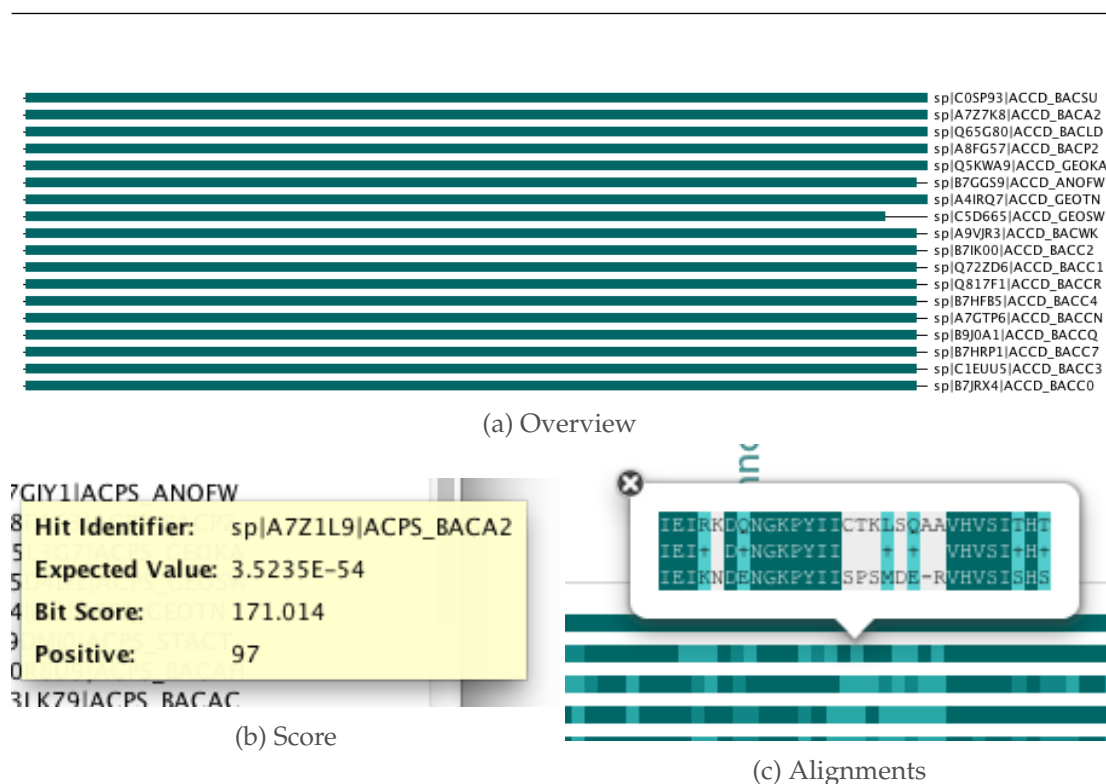


Figure B.2: Sequence homology observations for the beta subunit of acetyl-coenzyme A carboxylase (BG13926^[252])

Transport reactions A list of transport reactions for common metabolites is displayed for selection and inclusion in reconstructions. The list of reactions includes the creations of proton symport for amino acids, nucleobases and nucleosides. ATP binding cassette transporters can be created for amino acids, carbohydrates, minerals and ions. Transport of carbohydrates via the phosphotransferase system are not listed as the process may be encoded as one or two reactions.

Structure search Metabolite structures can be searched by drawing a query in the JChemPaint editor or by specifying a SMARTS input. Matching metabolites are selected and added as a new entity collection in the side bar (Fig. 3.8).

Sequence annotation A local sequence homology search using BLAST^[253] can be run on gene product entries. The homology search is started as a task that is queued and run in the background. Hits identified are loaded as observations on the gene products and displayed in the inspector (Fig. B.2a). The results can optionally contain the sequence alignment or just the scores (Fig. B.2b) and alignments are displayed

and coloured by conservation (Fig. [B.2c](#)). If the sequence was from UniProt, cross-references can be transferred from the observation to the gene product using the service framework.

Appendix C

Chapter 4 supplementary information

C.1 Introduction to graph theory

The algorithms used for ring perception are not specific to chemical structures and require several formal definitions. The basic concepts for these are briefly introduced here. A graph is composed of a set of vertices V and a set of edges E . Each vertex or edge may be *labelled* with a value. Two vertices are *adjacent* if an edge exists which contains the two vertices. The vertices of an edge are known as the endpoints, each endpoint is said to be *incident* to the edge. A *degree* of a vertex is the number of incident edges. If the endpoints are unordered, an edge is said to be *undirected*. Simple graphs have no edges connecting the same vertex (loops) and no edges which share the same endpoints (multiedges). We model a chemical structure as simple undirected labelled graph where the atoms and bonds are labels on the vertices and edges. Although the edges have a numeric value (bond order) they are not treated as weighted.

A *walk* is a sequence of vertices and edges connecting two vertices. If the start and end of the walk are the same, the walk is *closed*. Otherwise the walk is *open*. A *walk* is *simple* if it contains no repeated edges and *elementary* if there are no repeated vertices. A *simple* walk that is also *open* it is referred to as a *path*. Two vertices are *connected* if there is a *path* between them. A graph is *connected* if each vertex can be reached from every other vertex. A *connected component* ($\text{ConnComp}(G)$) in an undirected graph is a subgraph in which every vertex is *connected*.

A *cycle* is a closed *walk*. Graphs containing a *cycle* are said to be *cyclic* or *acyclic* if no cycle is present. Acyclic simple graphs are referred to as a *tree*. A ring in a chemical structure is best described as an *elementary* cycle. The cycle has no repeating vertices or edges and each vertex has a *degree* of 2 (in the cycle). This definition includes envelope rings of structures like naphthalene and azulene. As we are primarily concerned with chemical structures herein we use the term *cycle* to refer to *elementary cycle*.

A *cycle basis* is a set of cycles which can be used to generate all other cycles (cycle space) of the graph. Representing a cycle as a set of edges, a new cycle can be generated using the symmetric difference (XOR, \oplus -summing) of the edge sets of two cycles whose edge sets intersect. A *minimum cycle basis* is a cycle basis of minimum weight, in an unweighted graph the weight is simply the number of edges. When there is more than one basis with the same weight the choice between them is arbitrary as either can be used to generate the cycle space.

C.2 Determining cycle membership with a spanning tree

The SpanningTree was introduced in the CDK to eliminate acyclic vertices and edges, reducing the runtime of existing algorithms^[201]. A graph H is a subgraph of a graph G if the vertices V and edges E of H are a subset of G . A subgraph G is said to be a *spanning subgraph* of H if every vertex of H is present in G . The edges in chemical structures are unweighted and so the minimum spanning tree is a tree with the smallest number of edges. Given an input structure a spanning tree is created which contains a subset of the edges that span the vertices but contains no cycles. The SpanningTree class uses a greedy algorithm^[202] to sequentially build up this tree. Cyclic vertices and edges are determined by finding a path in the tree between the two endpoints of an edge which was not included. Any edge that is not in the spanning tree is cyclic and any path in the tree which connects the two endpoints contains vertices and edges that are also cyclic. The number of paths to find depends on the number of edges not included in the spanning tree. Structures containing a large number of rings will have more edges removed and more paths to find. Discovery of a path in the tree is implemented as depth-first-search and the entire tree may be traversed for each removed edge.

C.3 Drawbacks of using a timeout with AllRingsFinder

One major drawback of the existing implementation is the dependence on a time measure to determine feasibility. The time was measured from when the algorithm started and aborted if the elapsed time exceeded a set threshold. Whether the algorithm completes then depends on the machine specification and also the current load on the processor. The *timeout* was also generally left at a value too high (5 seconds) for larger datasets. To demonstrate this the timeout threshold was varied and tested on a small dataset. The number of structures that the algorithm successfully completed was measured. Increasing the threshold to longer than a second provides only a small gain in coverage (Fig. C.1). A timeout of just 50 ms allowed 99.4% of the structures to complete in 32 seconds. Leaving the timeout at the default value of 5000 ms allowed 99.8% of structures to complete but took nearly 10 times longer (291 seconds) (Fig. C.2). This could be an artefact of hardware improvements but highlights the difficulties in choosing an appropriate value when using a timeout. The set of all cycles was used throughout the library in fingerprint generation, similarity searching^[238], descriptors, kekulisation and fragmentation. The cycles were also partially utilised in aromaticity perception.

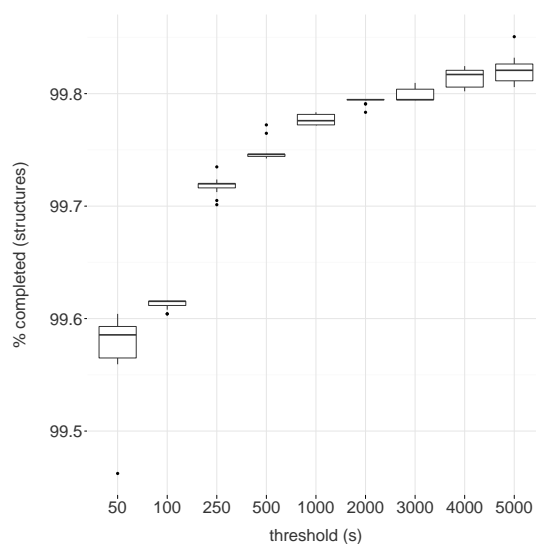


Figure C.1: Coverage of different timeout thresholds when finding all elementary cycles. The percentage of structures in ChEBI 108^[44] that the AllRingsFinder successfully finished was measured for different timeout thresholds. Increasing the threshold to more than a second has minimal impact.

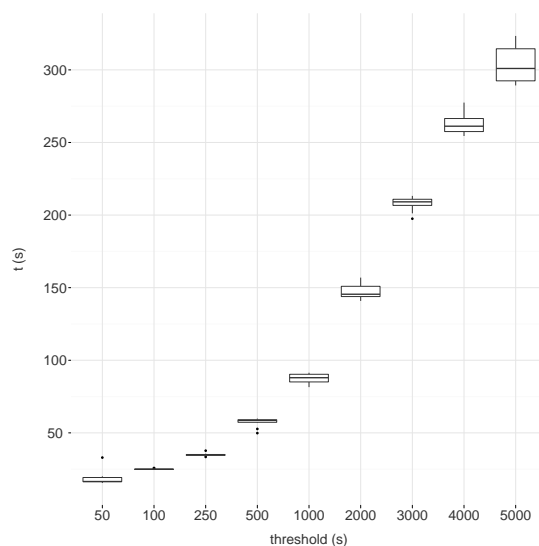


Figure C.2: Time taken of different timeout thresholds when finding all elementary cycles. The time taken for ChEBI 108^[44] to be processed by the AllRingsFinder was measured for different timeout thresholds.

C.4 Performance, absolute time taken

C.4.1 Cycle membership

Chemical structure	SpanningTree			RingSearch		
	mean t (ms)	median t (ms)	sdev	mean t (ms)	median t (ms)	sdev
chebi_108	2,969	2,826	246	236	212	90
nci_aug00	8,440	8,451	59	1,396	1,372	88
zinc_frag	5,338	5,353	77	1,833	1,818	60
chembl_17	122,357	122,493	616	10,325	10,303	98
zinc_leads	114,710	115,067	837	27,496	27,502	215

Times exclude IO but include conversion. It should be noted that the majority of the time spent in RingSearch was for the conversion (*adjacency*). Unprocessed measurements are provided in Additional file 1 - Cycle Membership Benchmark

C.4.2 Minimum cycle basis

Table C.1: Average ($n = 15$) time taken to compute the minimum cycle basis (MCB) using the old and new implementations

Chemical structure	MCB (old)			MCB (new)		
	mean t (ms)	median t (ms)	sdev	mean t (ms)	median t (ms)	sdev
chebi_108	4,200	4,020	695	396	353	160
nci_aug00	34,762	34,330	1,685	2,193	2,125	211
zinc_frag	47,752	47,844	1,982	2,376	2,330	153
chembl_17	245,620	245,592	990	15,341	15,257	311
zinc_leads	-	-	-	-	-	-

The time taken by the new implementation includes *adjacency* conversion (Tab. 4.1). Unprocessed measurements are provided in Additional file 2 - Short Cycles.

C.4.3 Relevant cycles

Table C.2: Average ($n = 15$) time taken to compute the relevant cycles using the old and new implementations

Chemical structure	Relevant cycles (old)			Relevant cycles (new)		
	mean t (ms)	median t (ms)	sdev	mean t (ms)	median t (ms)	sdev
chebi_108	17,013	16,897	576	445	388	171
nci_aug00	149,210	148,231	12,190	2,250	2,187	195
zinc_frag	183,587	184,720	18,219	2,610	2,519	237
chembl_17	972,493	972,605	1,197	16,003	15,991	201
zinc_leads	-	-	-	-	-	-

The time taken by the new implementation includes *adjacency* conversion (Tab. 4.1). Unprocessed measurements are provided in Additional file 2 - Short Cycles.

C.4.4 Essential cycles

Table C.3: Average ($n = 15$) time taken to compute the essential cycles using the old and new implementations

Chemical structure	Essential cycles (old)			Essential cycles (new)		
	mean t (ms)	median t (ms)	sdev	mean t (ms)	median t (ms)	sdev
chebi_108	16,561	16,395	615	572	424	451
nci_aug00	128,963	128,663	2,325	2,536	2,459	362
zinc_frag	217,312	217,016	940	3,662	3,574	336
chembl_17	954,954	952,437	20,698	17,235	17,171	293
zinc_leads	-	-	-	-	-	-

The time taken by the new implementation includes *adjacency* conversion (Tab. 4.1). Unprocessed measurements are provided in Additional file 2 - Short Cycles.

C.4.5 All elementary cycles

Table C.4: Average ($n = 15$) time taken to find all rings using old and new implementations of AllRingsFinder

Chemical structure	Old			New		
	t (ms)	median t (ms)	sdev	t (ms)	median t (ms)	sdev
chebi_108	16,293	16,179	540	599	574	105
nci_aug00	80,417	80,339	319	3,478	3,449	143
zinc_frag	41,854	41,747	458	3,786	3,778	60
chembl_17	568,984	568,809	829	25,200	25,181	99
zinc_leads	661,028	661,368	1133	54,490	54,471	101

The newer code includes the overhead of conversion to (and from) the CDK objects. (Tab. 4.1). Unprocessed measurements are provided in Additional file 3 - All Cycles

C.5 Feasibility of all elementary cycles

C.5.1 Number of cycles found

Table C.5: Average ($n = 15$) number of all cycles found in each datasets

Chemical structure	n structures	Old		New	
		Cycles	sdev	Cycles	sdev
chebi_108	26,790	98,597	199	126,713	0
nci_aug00	250,172	936,625	409	1,007,643	0
zinc_frag	504,074	1,022,498	0	1,022,498	0
chembl_17	1,318,180	6,176,585	378	6,599,942	0
zinc_leads	5,135,179	14,816,752	0	14,816,752	0

Due to the time out the old implementation sometimes provided different counts for each repeat.

C.5.2 Number of infeasible structures

Table C.6: Average ($n = 15$) number of structures considered infeasible by the old and new implementations

Chemical structure	n structures	n fail (old)	n fail (new)
chebi_108	26,790	108-117	41
nci_aug00	250,172	306-311	37
zinc_frag	504,074	0	0
chembl_17	1,318,180	2528-2547	232
zinc_leads	5,135,179	0	0

Based on a timeout procedure the number of cycles found in the old procedure varied between repeats, given the same structures the new feasibility threshold does not.

Appendix D

Chapter 5 supplementary information

D.1 Alternative aromaticity algorithms

Some aromaticity algorithms delocalise envelope ring by simply adding the count of the two smaller rings. This is sufficient when a single bridging bond is present but can be lead to strange results. The contrived example in Figure D.1 demonstrates the effect.

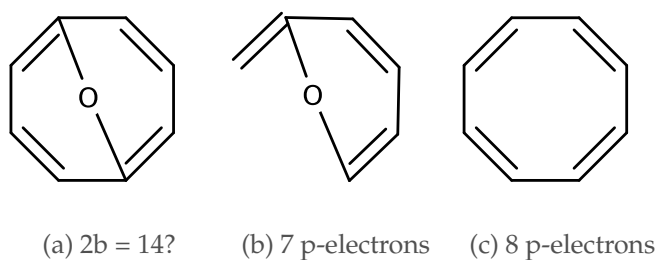


Figure D.1: O1C2=CC=CC1=CC=C2: If the electrons from the two small rings (D.1b) are summed directly, the system (D.1a) is encoded as aromatic.

D.2 Backtracking tautomer assignment

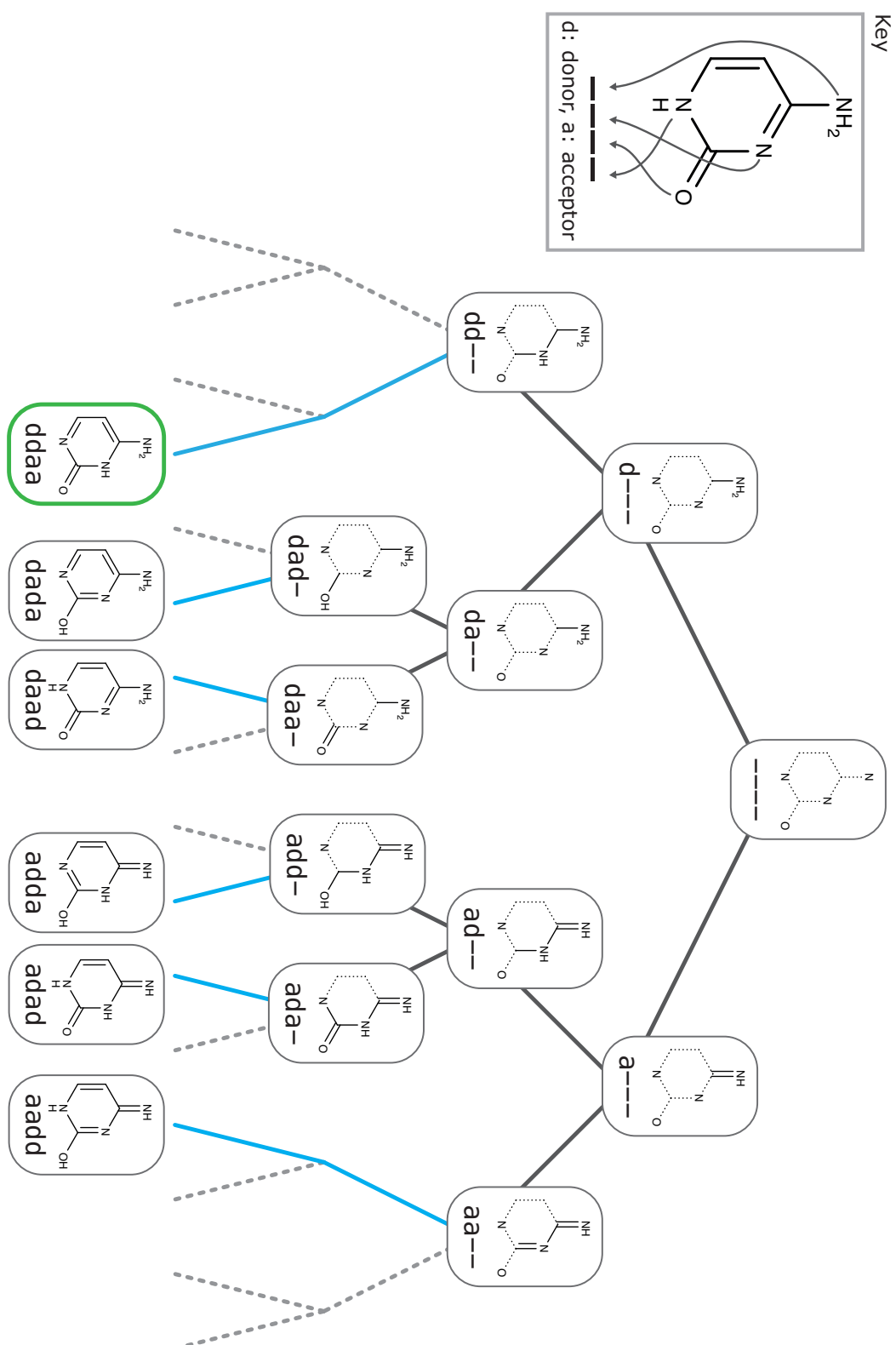


Figure D.2: Backtracking generation of cytosine tautomers, atom roles are initially unset (root). Atoms are assigned as a donor (left branches) or an acceptor (right branches). When all donors or acceptors have been assigned, the state can jump (blue lines) to the leaf. Dashed branches are not explored. The leaf highlighted in green is the first generated tautomer for this atom ordering. Branches are also not explored if an inconsistency is found but in this case all combinations of hydrogen placements are feasible.

Appendix E

Chapter 6 supplementary information

E.1 Verifying stereocentre comparison

Comparison of stereocentre configurations is tested by enumerating all possible orderings of the local representations. The following examples are taken from the Java class <https://github.com/cdk/cdk/blob/master/tool/smarts/src/test/java/org/openscience/cdk/isomorphism/SubstructureTest.java>.

Listing E.1: Examples of SMILES with the same tetrahedral configuration

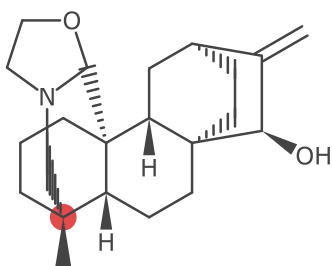
```
[C@](C)(N)(O)CC [C@](C)(O)(CC)N [C@](C)(CC)(N)(O) [C@@](C)(O)(N)CC  
[C@@](C)(CC)(O)N [C@@](C)(N)(CC)(O) [C@](N)(O)(C)CC [C@](N)(CC)(O)C  
[C@](N)(C)(CC)O [C@@](N)(C)(O)CC [C@@](N)(O)(CC)C [C@@](N)(CC)(C)O  
[C@](O)(CC)(C)N [C@](O)(N)(CC)C [C@](O)(C)(N)(CC) [C@@](O)(C)(CC)N  
[C@@](O)(CC)(N)C [C@@](O)(N)(C)(CC) [C@](CC)(C)(O)N [C@](CC)(N)(C)O  
[C@](CC)(O)(N)C [C@@](CC)(O)(C)N [C@@](CC)(C)(N)O [C@@](CC)(N)(O)C
```

Listing E.2: Examples of SMILES with the same geometric configuration

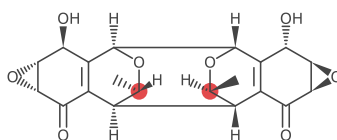
```
F/C=C/F F\C=C\F F/C(/[H])=C/F FC(/[H])=C/F F/C=C([H])/F  
F/C=C([H])F FC(/[H])=C([H])F C(\F)=C/F C(/F)=C\F
```

E.2 Differences in tetrahedral perception

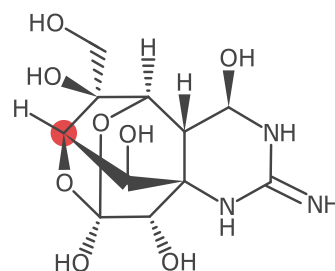
Highlighted atoms had a different configuration from the CDK 1.5 perception and the InChI. Invalid and ambiguous entries have now been updated by the ChEBI curators, the structures depicted here were those used in the analysis.



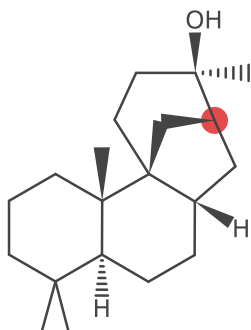
CHEBI:2909
invalid



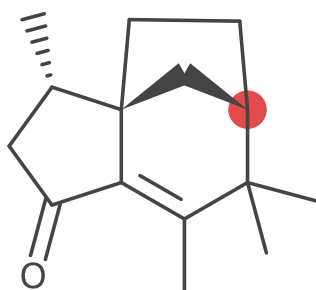
CHEBI:32956
invalid



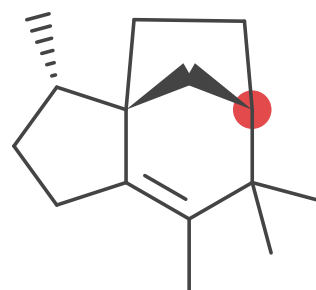
CHEBI:9506
invalid



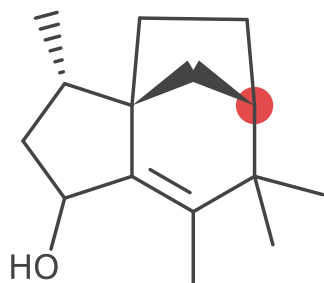
CHEBI:29519
ambiguous



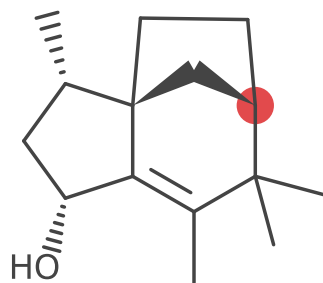
CHEBI:51460
ambiguous



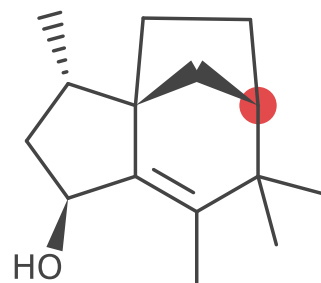
CHEBI:51458
ambiguous



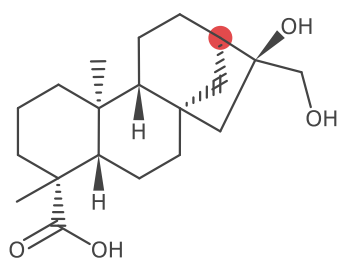
CHEBI:51478
ambiguous



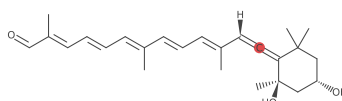
CHEBI:51479
ambiguous



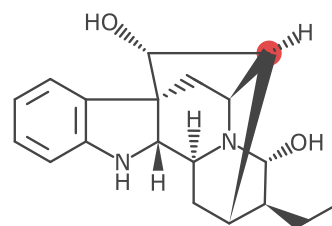
CHEBI:51480
ambiguous



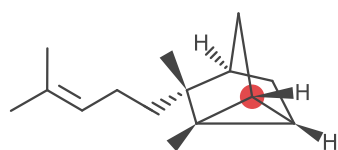
CHEBI:65778
ambiguous



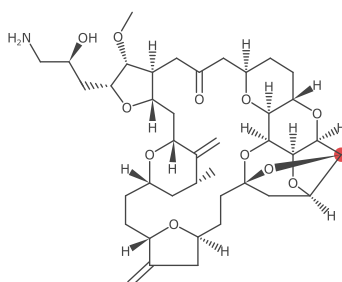
CHEBI:34596
missed by InChI



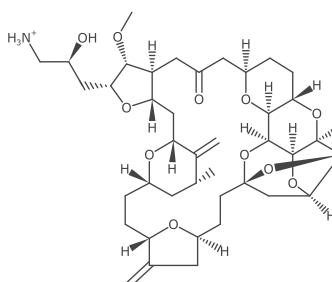
CHEBI:7621
abnormal bond lengths



CHEBI:61677
abnormal bond lengths



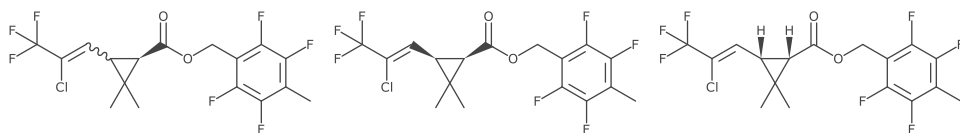
CHEBI:63587
abnormal bond lengths



CHEBI:70711
abnormal bond lengths

E.3 Unambiguous representation of tefluthrin

The ChEBI entry for tefluthrin ([CHEBI:9430](#)) was identified as ambiguous during evaluation of stereochemistry perception. The wavy (up/down) bond was located both between a geometric and tetrahedral stereocentre. The structure has now been split into two separate entries by the ChEBI curators.



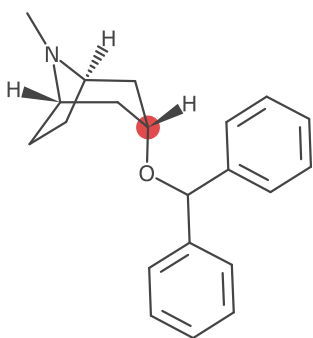
(a) [CHEBI:9430](#)

(b) [CHEBI:39395](#)

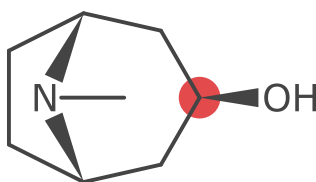
(c) [CHEBI:78106](#)

E.4 Differences in Cahn-Ingold-Prelog

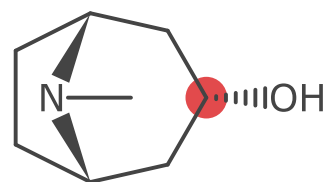
Structural depictions of ChEBI entries with a disagreement between CIP stereo label of highlighted atom.



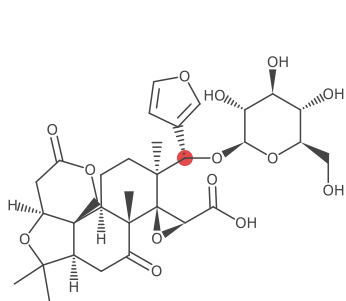
[CHEBI:3048](#)



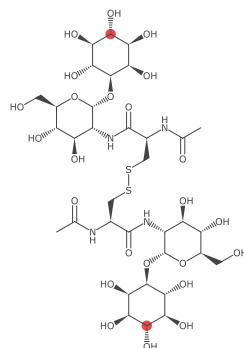
[CHEBI:15742](#)



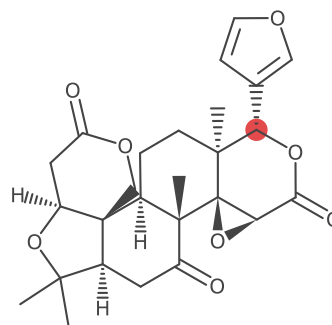
[CHEBI:15884](#)



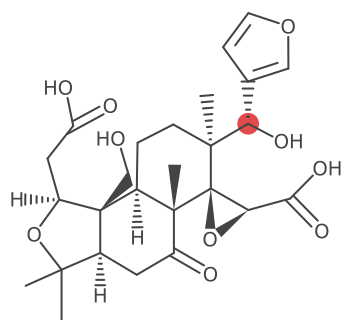
CHEBI:16063



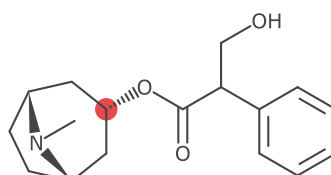
CHEBI:16086



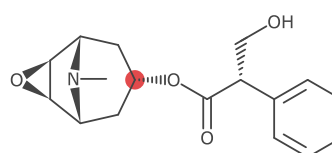
CHEBI:16226



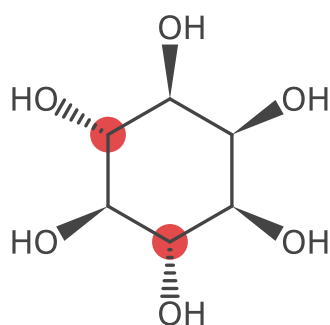
CHEBI:16419



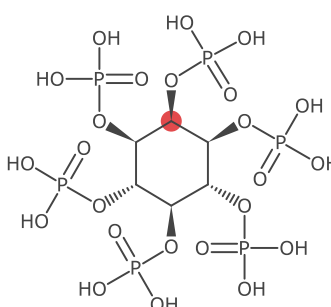
CHEBI:16684



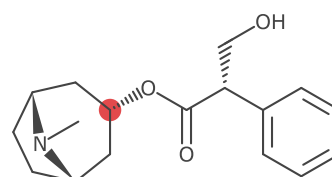
CHEBI:16794



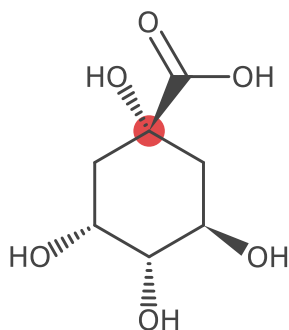
CHEBI:17268



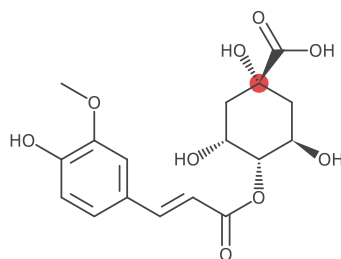
CHEBI:17401



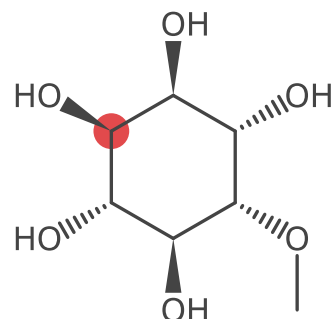
CHEBI:17486



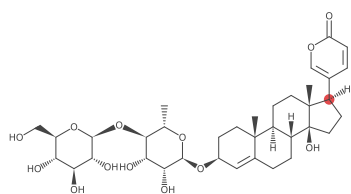
CHEBI:17521



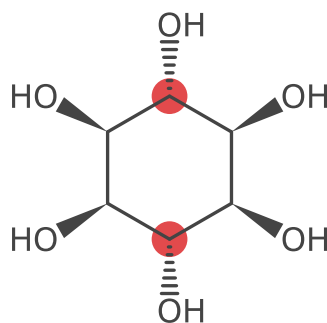
CHEBI:18013



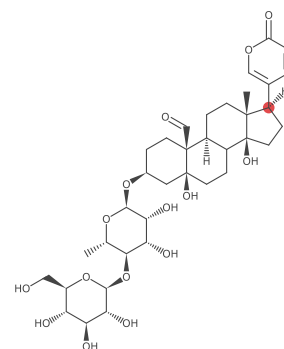
CHEBI:18173



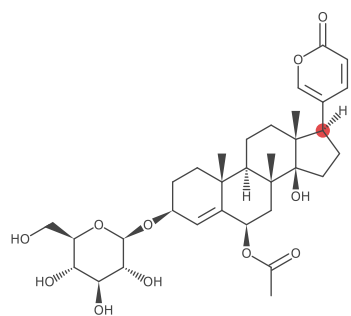
CHEBI:27831



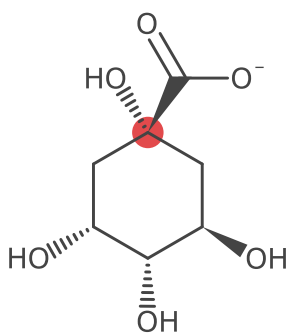
CHEBI:27987



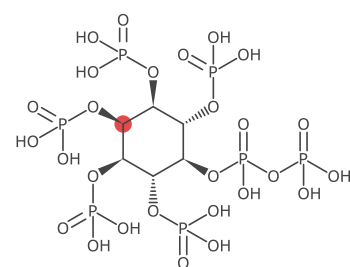
CHEBI:28271



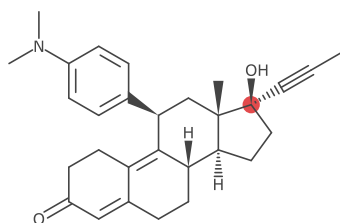
CHEBI:28332



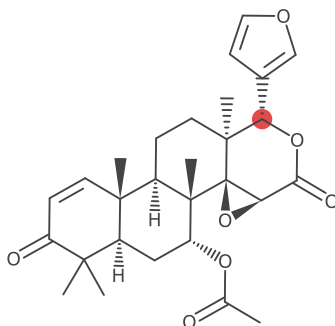
CHEBI:29751



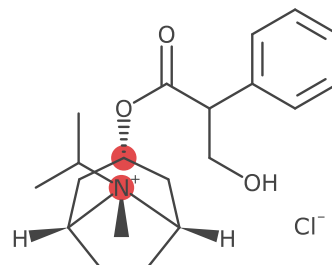
CHEBI:30164



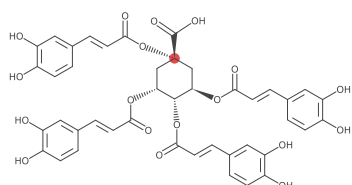
CHEBI:50692



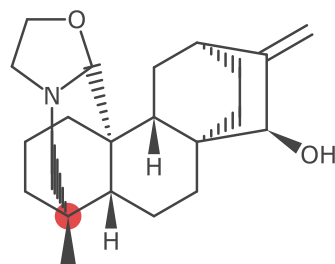
CHEBI:65954



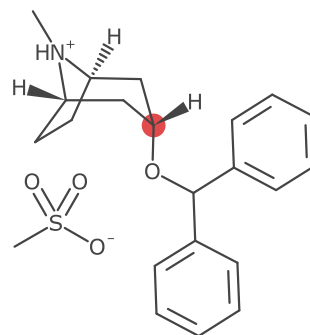
CHEBI:553542



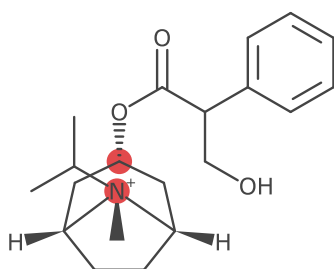
CHEBI:512



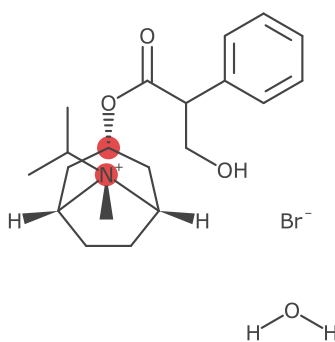
CHEBI:2909



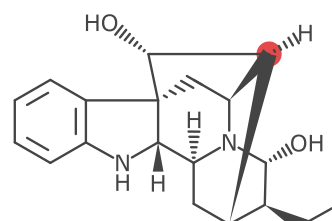
CHEBI:3049



CHEBI:5956



CHEBI:5957



CHEBI:7621

References

- [1] Monk J., Nogales J., and Palsson B. O. Optimizing genome-scale network reconstructions. *Nature Biotechnology*, 32(5):447–452, 2014. [1](#)
- [2] Feist A., Herrgård M., Thiele I., Reed J., and Palsson B. Reconstruction of biochemical networks in microorganisms. *Nat Rev Micro*, 7(2):129–143, 2009. [1](#), [87](#)
- [3] Bordbar A., Monk J., King Z., and Palsson B. O. Constraint-based models predict metabolic and associated cellular functions. *Nature Reviews*, 15:108–120, 2014. [1](#)
- [4] Feist A. M. and Palsson B. The growing scope of applications of genome-scale metabolic reconstructions using *Escherichia coli*. *Nature Biotechnology*, 26(6):659–667, 2008. [1](#)
- [5] Blazeck J. and Alper H. Systems metabolic engineering: Genome-scale models and beyond. *Biotechnology Journal*, 5(7):647–659, 2010. [1](#)
- [6] Yim H., Haselbeck R., Niu W., Pujol-Baxley C., Burgard A., Boldt J., Khandurina J., Trawick J. D., Osterhout R. E., Stephen R., Estadilla J., Teisan S., Schreyer H. B., Andrae S., Yang T. H., Lee S. Y., Burk M. J., and Dien S. V. Metabolic engineering of *Escherichia coli* for direct production of 1,4-butanediol. *Nature Chemical Biology*, 7(7):1–8, 2011. [1](#)
- [7] Mendes P., Smallbone K., and Standford N. Kinetic modelling of large-scale metabolic networks. In: *Proceedings of the 9th International Conference on Computational Methods in Systems Biology*, pages 1–3, 2011. [2](#)
- [8] Stanford N. J., Lubitz T., Smallbone K., Klipp E., Mendes P., and Liebermeister W. Systematic construction of kinetic models from genome-scale metabolic networks. *PLoS ONE*, 8(11):e79195, 2013. [2](#)
- [9] Price N. D., Reed J. L., and Palsson B. Genome-scale models of microbial cells: evaluating the consequences of constraints. *Nat Rev Microbiol*, 2(11):886–97, 2004. [2](#)
- [10] Orth J. D., Thiele I., and Palsson B. What is flux balance analysis? *Nat Biotechnol*, 28(3):245–248, 2010. [2](#)
- [11] Thiele I. and Palsson B. A protocol for generating a high-quality genome-scale metabolic reconstruction. *Nature Protocols*, 5(1):93–121, 2010. [3](#), [5](#), [83](#)
- [12] Delcher A. L., Harmon D., Kasif S., White O., and Salzberg S. L. Improved microbial gene identification with GLIMMER. *Nucleic Acids Research*, 27(23):4636–4641, 1999. [3](#)
- [13] Borodovsky M. and McIninch J. GENMARK: parallel gene recognition for both DNA strands. *Computers & chemistry*, 17(2):123–133, 1993. [3](#)
- [14] McDowall J. and Hunter S. InterPro protein classification. *Methods Mol Biol*, 694:37–47, 2011. [3](#)
- [15] Ebrahim A., Lerman J. A., Palsson B., and Hyduke D. R. COBRApy: COnstraints-Based Reconstruction and Analysis for Python. *BMC Systems Biology*, 7(74), 2013. [3](#), [10](#)
- [16] Ferrari L. D., Aitken S., van Hemert J., and Goryanin I. EnzML: multi-label prediction of enzyme classes using InterPro signatures. *BMC Bioinformatics*, 13(61), 2012. [4](#)
- [17] Saier M. H., Yen M. R., Noto K., Tamang D. G., and Elkan C. The Transporter Classification Database: recent advances. *Nucleic Acids Research*, 37(Database):D274–D278, 2009. [4](#)
- [18] Thiele I. and Palsson B. Reconstruction annotation jamborees: a community approach to systems biology. *Molecular Systems Biology*, 6(361), 2010. [4](#)
- [19] Hanson A. D., Pribat A., Waller J. C., and Crécy-Lagard V. D. ‘Unknown’ proteins and ‘orphan’ enzymes: the missing half of the engineering parts list – and how to find it. *Biochem. J.*, 425(1):1–11, 2009. [4](#)
- [20] Horton P., Park K.-J., Obayashi T., Fujita N., Harada H., Adams-Collier C., and Nakai K. WoLF PSORT: protein localization predictor. *Nucleic Acids Research*, 35(Web Server):W585–W587, 2007. [4](#)
- [21] Kumar V., Dasika M., and Maranas C. Optimization based automated curation of metabolic reconstructions. *BMC Bioinformatics*, 8(212), 2007. [5](#), [19](#)

-
- [22] Kumar V. and Maranas C. GrowMatch: an automated method for reconciling in silico/in vivo growth predictions. *PLoS Comput Biol*, 5(3):e1000308, 2009. 5
 - [23] Cuellar A. A., Nielsen P. F., Bullivant D. P., and Hunter P. J. CellML 1.1 for the definition and exchange of biological models. In *Conf. Proc. 2003 IFAC Symposium on Modelling and Control in Biomedical Systems.*, pages 451–456, 2003. 5
 - [24] Hucka M., Finney A., Sauro H., and Bolouri H. Systems Biology Markup Language (SBML) Level 1: Structures and facilities for basic model definitions. pages 1–38, 2003. 5, 83
 - [25] Demir E., Cary M. P., Paley S., Fukuda K., Lemer C., Vastrik I., Wu G., D'Eustachio P., Schaefer C., Luciano J., Schacherer F., Martinez-Flores I., Hu Z., Jimenez-Jacinto V., Joshi-Tope G., Kandasamy K., Lopez-Fuentes A. C., Mi H., Pichler E., Rodchenkov I., Splendiani A., Tkachev S., Zucker J., Gopinath G., Rajasimha H., Ramakrishnan R., Shah I., Syed M., Anwar N., Babur O., Blinov M., Brauner E., Corwin D., Donaldson S., Gibbons F., Goldberg R., Hornbeck P., Luna A., Murray-Rust P., Neumann E., Reubenacker O., Samwald M., van Iersel M., Wimalaratne S., Allen K., Braun B., Whirl-Carrillo M., Cheung K.-H., Dahlquist K., Finney A., Gillespie M., Glass E., Gong L., Haw R., Honig M., Hubaut O., Kane D., Krupa S., Kutmon M., Leonard J., Marks D., Merberg D., Petri V., Pico A., Ravenscroft D., Ren L., Shah N., Sunshine M., Tang R., Whaley R., Letovksy S., Buetow K. H., Rzhetsky A., Schachter V., Sobral B. S., Dogrusoz U., McWeeney S., Aladjem M., Birney E., Collado-Vides J., Goto S., Hucka M., Novère N. L., Maltsev N., Pandey A., Thomas P., Wingender E., Karp P. D., Sander C., and Bader G. D. The BioPAX community standard for pathway data sharing. *Nat Biotechnol*, 28(9): 935–942, 2010. 5, 95
 - [26] Bergmann F., Shapiro B., and Hucka M. SBML software matrix. online, 2014. URL http://sbml.org/SBML_Software_Guide/SBML_Software_Matrix. accessed May 2014. 6
 - [27] Juty N., Novère N. L., and Laibe C. Identifiers.org and MIRIAM Registry: community resources to provide persistent identification. *Nucleic Acids Research*, 40(D1):D580–D586, 2012. 6, 18, 83
 - [28] Li C., Donizelli M., Rodriguez N., Dharuri H., Endler L., Chelliah V., Li L., He E., Henry A., and Stefan M. BioModels Database: An enhanced, curated and annotated resource for published quantitative kinetic models. *BMC Systems Biology*, 4(92), 2010. 6
 - [29] Schellenberger J., Park J. O., Conrad T. M., and Palsson B. BiGG: a biochemical genetic and genomic knowledgebase of large scale metabolic reconstructions. *BMC Bioinformatics*, 11(213), 2010. 7
 - [30] Kanehisa M. The KEGG resource for deciphering the genome. *Nucleic Acids Research*, 32(Database issue):D277–D280, 2004. 7
 - [31] Kanehisa M., Goto S., Sato Y., Furumichi M., and Tanabe M. KEGG for integration and interpretation of large-scale molecular data sets. *Nucleic Acids Research*, 40(D1):D109–D114, 2012. 7
 - [32] SRI International. Biocyc. online. URL <http://biocyc.org/>. accessed May 2014. 7
 - [33] Karp P., Ouzounis C., Moore-Kochlacs C., Goldovsky L., Kaipa P., Ahrén D., Tsoka S., Darzentas N., Kunin V., and López-Bigas N. Expansion of the BioCyc collection of Pathway/Genome Databases to 160 genomes. *Nucleic Acids Research*, 33(19):6083–6089, 2005. 7
 - [34] Caspi R., Altman T., Billington R., Dreher K., Foerster H., Fulcher C. A., Holland T. A., Keseler I. M., Kothari A., Kubo A., Krummenacker M., Latendresse M., Mueller L. A., Ong Q., Paley S., Subhraveti P., Weaver D. S., Weerasinghe D., Zhang P., and Karp P. D. The MetaCyc database of metabolic pathways and enzymes and the BioCyc collection of Pathway/Genome Databases. *Nucleic Acids Research*, 42(D1):D459–D471, 2014. 7, 46
 - [35] Croft D., O’Kelly G., Wu G., Haw R., Gillespie M., Matthews L., Caudy M., Garapati P., Gopinath G., and Jassal B. Reactome: a database of reactions, pathways and biological processes. *Nucleic Acids Research*, 39(suppl 1):D691–D697, 2011. 7
 - [36] Morgat A., Coissac E., Coudert E., Axelsen K., Keller G., Bairoch A., Bridge A., Bougueleret L., Xenarios I., and Viari A. UniPathway: a resource for the exploration and annotation of metabolic pathways. *Nucleic Acids Research*, 40(D1):D761–D769, 2012. 7
 - [37] Kelder T., van Iersel M., Hanspers K., Kutmon M., Conklin B., Evelo C., and Pico A. WikiPathways: building research communities on biological pathways. *Nucleic Acids Research*, 40(D1):D1301–D1307, 2012. 7
 - [38] Bairoch A. The ENZYME database in 2000. *Nucleic Acids Research*, 28(1):304–305, 2000. 7
 - [39] Fleischmann A., Darsow M., Degtyarenko K., Fleischmann W., Boyce S., Axelsen K., Bairoch A., Schomburg D., Tipton K., and Apweiler R. IntEnz, the integrated relational enzyme database. *Nucleic Acids Research*, 32(suppl 1):D434–D437, 2004. 7
 - [40] Scheer M., Grote A., Chang A., Schomburg I., Munaretto C., Rother M., Sohngen C., Stelzer M., Thiele J., and Schomburg D. BRENDA, the enzyme information system in 2011. *Nucleic Acids Research*, 39(Database):D670–D676, 2011. 7
 - [41] Chang A., Scheer M., Grote A., Schomburg I., and Schomburg D. BRENDA, AMENDA and FRENDA the enzyme information system: new content and tools in 2009. *Nucleic Acids Research*, 37(suppl 1):D588–D592, 2009. 7

-
- [42] Alcántara R., Axelsen K., Morgat A., Belda E., Coudert E., Bridge A., Cao H., Matos P. D., Ennis M., and Turner S. Rhea—a manually curated resource of biochemical reactions. *Nucleic Acids Research*, 40(D1):D754–D760, 2012. 7
- [43] Ren Q., Chen K., and Paulsen I. T. TransportDB: a comprehensive database resource for cytoplasmic membrane transport systems and outer membrane channels. *Nucleic Acids Research*, 35(Database):D274–D279, 2007. 7
- [44] Hastings J., Matos P. D., Dekker A., Ennis M., Harsha B., Kale N., Muthukrishnan V., Owen G., Turner S., Williams M., and Steinbeck C. The ChEBI reference database and ontology for biologically relevant chemistry: enhancements for 2013. *Nucleic Acids Research*, 41(D1):D456–D463, 2013. 8, 135, 241, 242
- [45] Wishart D. S., Jewison T., Guo A. C., Wilson M., Knox C., Liu Y., Djoumbou Y., Mandal R., Aziat F., Dong E., Bouatra S., Sinelnikov I., Arndt D., Xia J., Liu P., Yallou F., Bjorn Dahl T., Perez-Pineiro R., Eisner R., Allen F., Neveu V., Greiner R., and Scalbert A. HMDB 3.0 –the human metabolome database in 2013. *Nucleic Acids Research*, 41(D1):D801–D807, 2013. 8, 108
- [46] Sud M., Fahy E., Cotter D., Brown A., Dennis E. A., Glass C. K., Merrill A. H., Murphy R. C., Raetz C. R. H., Russell D. W., and Subramaniam S. LMSD: LIPID MAPS structure database. *Nucleic Acids Research*, 35(Database):D527–D532, 2007. 8
- [47] Foster J. M., Moreno P., Fabregat A., Hermjakob H., Steinbeck C., Apweiler R., Wakelam M. J. O., and Vizcaino J. A. LipidHome: A database of theoretical lipids optimized for high throughput mass spectrometry lipidomics. *PLoS ONE*, 8(5):e61951, 2013. 8
- [48] Wohlgemuth G., Haladiya P. K., Willighagen E., Kind T., and Fiehn O. The Chemical Translation Service – a web-based tool to improve standardization of metabolomic reports. *Bioinformatics*, 26(20):2647–2648, 2010. 8, 29
- [49] Lee T., Pouliot Y., Wagner V., Gupta P., Stringer-Calvert D., Tenenbaum J., and Karp P. D. BioWarehouse: a bioinformatics database warehouse toolkit. *BMC Bioinformatics*, 7(170), 2006. 8
- [50] Lang M., Stelzer M., and Schomburg D. BKM-react, an integrated biochemical reaction database. *BMC Biochemistry*, 12(42), 2011. 8, 29
- [51] Kumar A., Suthers P., and Maranas C. MetRxn: a knowledgebase of metabolites and reactions spanning metabolic models and databases. *BMC Bioinformatics*, 13(6), 2012. 8, 29
- [52] Bernard T., Bridge A., Morgat A., Moretti S., Xenarios I., and Pagni M. Reconciliation of metabolites and biochemical reactions for metabolic networks. *Briefings in Bioinformatics*, 15(1):123–135, 2014. 8, 29
- [53] Bolton E., Wang Y., Thiessen P., and Bryant S. PubChem: Integrated platform of small molecules and biological activities. In *Annual Reports in Computational Chemistry*, volume 4, chapter 12. American Chemical Society, Washington, 2008. 8, 122
- [54] Pence H. and Williams A. ChemSpider: an online chemical information resource. *Journal of Chemical Education*, 87(11):1123–1124, 2010. 8
- [55] Schilling C. H., Kane S., Roth M., Ruan J., Stadskev K., Thakar R., Travnik E., van Dien S., and Wiback S. SimPheny: A computation infrastructure for systems biology. 2005. 8
- [56] Karp P. D., Paley S., Krummenacker M., Latendresse M., Dale J. M., Lee T. J., Kaipa P., Gilham F., Spaulding A., Popescu L., Altman T., Paulsen I., Keseler I. M., and Caspi R. Pathway Tools version 13.0: integrated software for pathway/genome informatics and systems biology. *Briefings in Bioinformatics*, 11(1):40–79, 2010. 9, 84
- [57] Green M. L. and Karp P. D. A Bayesian method for identifying missing enzymes in predicted metabolic pathway databases. *BMC Bioinformatics*, 5(76), 2004. 9
- [58] Latendresse M., Krummenacker M., Trupp M., and Karp P. D. Construction and completion of flux balance models from pathway databases. *Bioinformatics*, 28(3):388–396, 2012. 9, 18
- [59] Henry C. S., DeJongh M., Best A. A., Frybarger P. M., Lindsay B., and Stevens R. L. High-throughput generation, optimization and analysis of genome-scale metabolic models. *Nature Biotechnology*, 28(9):969–974, 2010. 9
- [60] Aziz R. K., Bartels D., Best A. A., DeJongh M., Disz T., Edwards R. A., Formsma K., Gerdes S., Glass E. M., Kubal M., Meyer F., Olsen G. J., Olson R., Osterman A. L., Overbeek R. A., McNeil L. K., Paarmann D., Paczian T., Parrello B., Pusch G. D., Reich C., Stevens R., Vassieva O., Vonstein V., Wilke A., and Zagnitko O. The RAST Server: Rapid Annotations using Subsystems Technology. *BMC Genomics*, 9(75), 2008. 9
- [61] Swainston N., Smallbone K., Mendes P., Kell D., and Paton N. The SuBliMinaL Toolbox: automating steps in the reconstruction of metabolic networks. *J Integr Bioinform*, 8(186), 2011. 9, 30, 84
- [62] Swainston N. and Mendes P. libAnnotationSBML: a library for exploiting sbml annotations. *Bioinformatics*, 25(17):2292–2293, 2009. 9, 84
- [63] Herrgård M. J., Swainston N., Dobson P., Dunn W. B., Arga K. Y., Arvas M., Buthgen N., Borger S., Costenoble R., Heinemann M., Hucka M., Novère N. L., Li P., Liebermeister W., Mo M. L., Oliveira A. P., Petranovic D., Pettifer S., Simeonidis E., Smallbone K., Spasić I., Weichart D., Brent R., Broomhead D. S., Westerhoff H. V., Kürdar B., Penttilä M., Klipp E., Palsson B., Sauer U., Oliver S. G., Mendes P., Nielsen J., and Kell D. B. A consensus yeast metabolic network reconstruction obtained from a community approach to systems biology.
-

- Nat Biotechnol*, 26(10):1155–1160, 2008. [9](#), [21](#)
- [64] Thiele I, Swainston N., Fleming R., Hoppe A., Sahoo S., Aurich M., Haraldsdottir H., Mo M., Rolfsson O., and Stobbe M. A community-driven global reconstruction of human metabolism. *Nat Biotechnol*, 31(5):419–425, 2013. [9](#), [21](#), [28](#), [90](#), [106](#), [109](#), [111](#)
 - [65] Büchel F., Rodriguez N., Swainston N., Wrzodek C., Czauderna T., Keller R., Mittag F., Schubert M., Glont M., and Golebiewski M. Path2Models: large-scale generation of computational models from biochemical pathway maps. *BMC Systems Biology*, 7(116), 2013. [9](#)
 - [66] Agren R., Liu L., Shoaie S., Vongsangnak W., Nookaew I., and Nielsen J. The RAVEN toolbox and its use for generating a genome-scale metabolic model for *Penicillium chrysogenum*. *PLoS Comput Biol*, 9(3):e1002980, 2013. [10](#), [84](#)
 - [67] Ganter M., Bernard T., Moretti S., Stelling J., and Pagni M. MetaNetX.org: a website and repository for accessing, analysing and manipulating metabolic networks. *Bioinformatics*, 29(6):815–816, 2013. [10](#)
 - [68] Ganter M., Kaltenbach H.-M., and Stelling J. Predicting network functions with nested patterns. *Nature Communications*, 5(3006), 2014. [10](#)
 - [69] Becker S., Feist A., Mo M., Hannum G., Palsson B., and Herrgard M. Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox. *Nature Protocols*, 2(3):727–738, 2007. [10](#)
 - [70] Schellenberger J., Que R., Fleming R., Thiele I., Orth J., Feist A., Zielinski D., Bordbar A., Lewis N., and Rahmann S. Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox v2.0. *Nature Protocols*, 6(9):1290–1307, 2011. [10](#)
 - [71] Thorleifsson S. G. and Thiele I. rBioNet: A COBRA toolbox extension for reconstructing high-quality biochemical networks. *Bioinformatics*, 27(14):2009–2010, 2011. [10](#)
 - [72] Sun J. and Zeng A.-P. IdentiCS – identification of coding sequence and in silico reconstruction of the metabolic network directly from unannotated low-coverage bacterial genome sequence. *BMC Bioinformatics*, 5(112), 2004. [10](#)
 - [73] Arakawa K., Yamada Y., Shinoda K., Nakayama Y., and Tomita M. GEM System: automatic prototyping of cell-wide metabolic pathway models from genomes. *BMC Bioinformatics*, 7(168), 2006. [10](#)
 - [74] Forth T., McConkey G. A., and Westhead D. R. MetNetMaker: a free and open-source tool for the creation of novel metabolic networks in SBML format. *Bioinformatics*, 26(18):2352–2353, 2010. [10](#)
 - [75] Dias O., Rocha M., Ferreira E., and Rocha I. Merlin: Metabolic models reconstruction using genome-scale information. In *11th International Symposium on Computer Applications in Biotechnology 11th International Symposium on Computer Applications in Biotechnology 11th International Symposium on Computer Applications in Biotechnology 11th International Symposium on Computer Applications in Biotechnology*, Leuven, Belgium, 2010. [10](#)
 - [76] Liao Y.-C., Tsai M.-H., Chen F.-C., and Hsiung C. A. GEMSiRV: a software platform for genome-scale metabolic model simulation, reconstruction and visualization. *Bioinformatics*, 28(13):1752–1758, 2012. [11](#), [85](#)
 - [77] Liao Y.-C., Chen J., Tsai M.-H., Tang Y.-H., Chen F.-C., and Hsiung C. MrBac: a web server for draft metabolic network reconstructions for bacteria. *Bioeng Bugs*, 2:284–287, 2011. [11](#)
 - [78] Boele J., Olivier B. G., and Teusink B. FAME, the Flux Analysis and Modeling Environment. *BMC Systems Biology*, 6(8), 2012. [11](#)
 - [79] Feng X., Xu Y., Chen Y., and Tang Y. MicrobesFlux: a web platform for drafting metabolic models from the KEGG database. *BMC Systems Biology*, 6(94), 2012. [11](#)
 - [80] Pabinger S., Snajder R., Hardiman T., Willi M., Dander A., and Trajanoski Z. MEMOSys 2.0: an update of the bioinformatics database for genome-scale models and genomic data. *Database*, 2014(0):bau004–bau004, 2014. [11](#)
 - [81] Rocha I., Maia P., Evangelista P., Vilaça P., Soares S., Pinto J., Nielsen J., Patil K., Ferreira E., and Rocha M. OptFlux: an open-source software platform for in silico metabolic engineering. *BMC Systems Biology*, 4(45), 2010. [11](#)
 - [82] Thiele I., Fleming R. M. T., Que R., Bordbar A., Diep D., and Palsson B. Multiscale modeling of metabolism and macromolecular synthesis in *E. coli* and its application to the evolution of codon usage. *PLoS ONE*, 7(9):e45635, 2012. [12](#)
 - [83] Gasteiger J., editor. *Handbook of Chemoinformatics*, volume 1. Wiley VCH, 2003. [12](#), [147](#)
 - [84] Faulon J.-L. and Bender A., editors. *Handbook of Chemoinformatics Algorithms*. Chapman and Hall/CRC, 2010. [12](#)
 - [85] Balaban A. T. *Chemical applications of graph theory*. Academic Press Inc, 1976. [12](#)
 - [86] Bauerschmidt S. and Gasteiger J. Overcoming the limitations of a connection table description: A universal representation of chemical species. *Journal of chemical information and computer sciences*, 37(4):705–714, 1997. [12](#)
 - [87] Dietz A. Yet another representation of molecular structure. *Journal of chemical information and computer sciences*, 35(5):787–802, 1995. [12](#)

-
- [88] McGregor J. Backtrack search algorithms and the maximal common subgraph problem. *Software: Practice and Experience*, 12(1):23–34, 1982. 13
- [89] Raymond J., Gardiner E., and Willett P. RASCAL: Calculation of graph similarity using maximum common edge subgraphs. *The Computer Journal*, 45(6):631–644, 2002. 13
- [90] Willett P. Similarity-based virtual screening using 2D fingerprints. *Drug discovery today*, 11(23):1046–1053, 2006. 13
- [91] Gao X., Xiao B., Tao D., and Li X. A survey of graph edit distance. *Pattern Anal Applic*, 13(1):113–129, 2010. 13
- [92] Bender A. and Glen R. Molecular similarity: a key technique in molecular informatics. *Org. Biomol. Chem.*, 2, 2(22):3204–3218, 2004. 13
- [93] Warr W. A. Representation of chemical structures. *WIREs Comput Mol Sci*, 1(4):557–579, 2011. 14
- [94] Murray-Rust P. and Rzepa H. S. Chemical Markup, XML, and the Worldwide Web. 1. basic principles. *J. Chem. Inf. Comput. Sci.*, 39(6):928–942, 1999. 14
- [95] Lowe D., Corbett P., Murray-Rust P., and Glen R. Chemical name to structure: OPSIN, an open source solution. *Journal of chemical information and modeling*, 51(3):739–753, 2011. 15, 110
- [96] Wisniewski J. L. AUTONOM: system for computer translation of structural diagrams into IUPAC-compatible names. 1. general design. *Journal of chemical information and computer sciences*, 30(3):324–332, 1990. 15
- [97] Accelrys. *CTfile Formats*. Accelrys, 2011. URL <http://accelrys.com/products/informatics/cheminformatics/ctfile-formats/no-fee.php>. 15, 148
- [98] Gushurst A., Nourse J., Hounshell W., Leland B., and Raich D. The substance module: the representation, storage, and searching of complex structures. *Journal of chemical information and computer sciences*, 31(4):447–454, 1991. 15
- [99] ECOTOX Support. SMILES tutorial. online, 2009. URL http://www.epa.gov/med/Prods_Pubs/smiles.htm. accessed May 2014. 15
- [100] Daylight CIS. Daylight Chemical Information Systems, Inc. online. URL <http://daylight.com/>. accessed May 2014. 15
- [101] Weininger D. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988. 15, 135
- [102] Weininger D., Weininger A., and Weininger J. SMILES. 2. Algorithm for generation of unique SMILES notation. *Journal of Chemical Information and Computer Sciences*, 29(2):97–101, 1989. 15, 31, 166
- [103] Daylight CIS. SMARTS – a language for describing molecular patterns. online, 2007. URL <http://www.daylight.com/dayhtml/doc/theory/theory.smarts.html>. accessed May 2014. 16
- [104] Heller S., Mcnaught A., Stein S., Tchekhovskoi D., and Pletnev I. InChI – the worldwide chemical structure identifier standard. *Journal of Cheminformatics*, 5(1):1–9, 2013. 16, 32
- [105] Pletnev I., Erin A., Mcnaught A., Blinov K., Tchekhovskoi D., and Heller S. InChIKey collision resistance: an experimental testing. *Journal of Cheminformatics*, 4(1):39, 2012. 16
- [106] Williams A. An InChIkey collision is discovered and not based on stereochemistry. online. URL <http://www.chemconnector.com/2011/09/01/an-inchikey-collision-is-discovered-and-not-based-on-stereochemistry/>. accessed May 2014. 16
- [107] Heller S. The status of the IUPAC International Chemical Identifier standard - InChI. In *10th International Conference on Chemical Structures and the 10th German Conference on Chemoinformatics*. URL <http://www.hellers.com/steve/pub-talks/ams-5-14.pdf>. 16, 33
- [108] Ihlenfeldt W. D., Takahashi Y., Abe H., and Sasaki S. Computation and management of chemical properties in CACTVS: An extensible networked approach toward modularity and compatibility. *J. Chem. Inf. Comp. Sci.*, 34(1):109–116, 1994. 17, 144
- [109] Steinbeck C., Han Y., Kuhn S., Horlacher O., Luttmann E., and Willighagen E. The Chemistry Development Kit (CDK): an open-source Java library for chemo- and bioinformatics. *J Chem Inf Comput Sci*, 43(2):493–500, 2003. 17
- [110] Steinbeck C., Hoppe C., Kuhn S., Floris M., Guha R., and Willighagen E. Recent developments of the Chemistry Development Kit (CDK) - an open-source Java library for chemo- and bioinformatics. *CPD*, 12(17):2111–2120, 2006. 17
- [111] O’Boyle N. M., Banck M., James C. A., Morley C., Vandermeersch T., and Hutchison G. R. Open Babel: An open chemical toolbox. *Journal of Cheminformatics*, 3(33), 2011. 17, 144
- [112] O’Boyle N., Guha R., Willighagen E., Adams S., Alvarsson J., Bradley J.-C., Filippov I., Hanson R., Hanwell M., and Hutchison G. Open Data, Open Source and Open Standards in chemistry: The Blue Obelisk five years on. *Journal of Cheminformatics*, 3(37), 2011. 17
- [113] Guha R. Chemical informatics functionality in R. *Journal of Statistical Software*, 18(5):1–16, 2007. 17
-

-
- [114] Harmon C. Cheminformatics, Java and, of Course, Common Lisp. online, 2013. URL <http://cyrusharmon.org/blog/display?id=121>. accessed May 2014. 17
- [115] Kuhn T., Willighagen E. L., Zielesny A., and Steinbeck C. CDK-Taverna: an open workflow environment for cheminformatics. *BMC Bioinformatics*, 11(159), 2010. 17
- [116] Truszkowski A., Jayaseelan K. V., Neumann S., Willighagen E. L., Zielesny A., and Steinbeck C. New developments on the cheminformatics open workflow environment CDK-Taverna. *Journal of Cheminformatics*, 3(54), 2011. 17
- [117] Beiskens S., Meinl T., Wiswedel B., de Figueiredo L., Berthold M., and Steinbeck C. KNIME-CDK: workflow-driven cheminformatics. *BMC Bioinformatics*, 14(257), 2013. 17
- [118] Feist A., Henry C., Reed J., Krummenacker M., Joyce A., Karp P., Broadbelt L., Hatzimanikatis V., and Palsson B. A genome-scale metabolic reconstruction for *Escherichia coli* K-12 MG1655 that accounts for 1260 ORFs and thermodynamic information. *Molecular Systems Biology*, 3(121), 2007. 19
- [119] Haraldsdóttir H., Thiele I., and Fleming R. Quantitative assignment of reaction directionality in a multicompartmental human metabolic reconstruction. *Biophysical Journal*, 102(8):1703–1711, 2012. 19, 29
- [120] Jankowski M. D., Henry C. S., Broadbelt L. J., and Hatzimanikatis V. Group contribution method for thermodynamic analysis of complex metabolic networks. *Biophysical Journal*, 95(3):1487–1499, 2008. 19
- [121] Noor E., Haraldsdóttir H. S., Milo R., and Fleming R. M. T. Consistent estimation of Gibbs energy using component contributions. *PLoS Comput Biol*, 9(7), 2013. 19
- [122] Rahman S. A., Cuesta S. M., Furnham N., Holliday G. L., and Thornton J. M. EC-BLAST: a tool to automatically search and compare enzyme reactions. *Nat Meth*, 11(2):171–174, 2014. 19, 21, 198
- [123] Latendresse M., Malerich J., Travers M., and Karp P. D. Accurate atom-mapping computation for biochemical reactions. *Journal of Chemical Information and Modeling*, 52(11):2970–2982, 2012. 84, 185
- [124] Heinonen M., Lappalainen S., Mielikäinen T., and Rousu J. Computing atom mappings for biochemical reactions without subgraph isomorphism. *Journal of Computational Biology*, 18(1):43–58, 2011. 19
- [125] Arita M. Introduction to the arm database: database on chemical transformations in metabolism for tracing pathways. In Tomita M. and Nishioka T., editors, *Metabolomics: the frontier of systems biology*, chapter 13, pages 193–210. Springer, 2005. 19
- [126] Zamboni N., Fendt S.-M., Rühl M., and Sauer U. ¹³C-based metabolic flux analysis. *Nat Protoc*, 4(6):878–892, 2009. 20
- [127] Ravikirthi P., Suthers P., and Maranas C. Construction of an *E. Coli* genome-scale atom mapping model for MFA calculations. *Biotechnol. Bioeng.*, 108(6):1372–1382, 2011. 20, 117
- [128] Arita M. Metabolic reconstruction using shortest paths. *Simulation Practice and Theory*, 8(1):109–125, 2000. 20
- [129] Pitkänen E., Jouhten P., and Rousu J. Inferring branching pathways in genome-scale metabolic networks. *BMC Syst Biol*, 3(103), 2009. 20
- [130] Blum T. and Kohlbacher O. Using atom mapping rules for an improved detection of relevant routes in weighted metabolic networks. *Journal of Computational Biology*, 15(6):565–576, 2008. 20
- [131] Latendresse M., Krummenacker M., and Karp P. D. Optimal metabolic route search based on atom mappings, 2014. Advanced access. 20, 84
- [132] Ugi I., Bauer J., Brandt J., Freidrich J., Gasteiger J., Jochum C., and Schubert W. New applications of computers in chemistry. *Angew. Chem. Int. Ed. Engl.*, 18(2):111–123, 1979. 20
- [133] Daylight CIS. SMIRKS - A Reaction Transform Language. online, 2007. URL <http://www.daylight.com/dayhtml/doc/theory/theory.smirks.html>. accessed May 2014. 20
- [134] Chanon M. and Barone R. Computer-Assisted Synthesis Design (CASD). In Gasteiger J., editor, *Handbook of Chemoinformatics*. Wiley VCH, 2003. 20
- [135] Hatzimanikatis V., Li C., Ionita J. A., Henry C. S., Jankowski M. D., and Broadbelt L. J. Exploring the diversity of complex metabolic networks. *Bioinformatics (Oxford, England)*, 21(8):1603–9, 2005. 21
- [136] Moriya Y., Shigemizu D., Hattori M., Tokimatsu T., Kotera M., Goto S., and Kanehisa M. PathPred: an enzyme-catalyzed metabolic pathway prediction server. *Nucleic Acids Research*, 38(Web Server):W138–W143, 2010. 21
- [137] Carbonell P., Planson A.-G., Fichera D., and Faulon J.-L. A retrosynthetic biology approach to metabolic pathway design for therapeutic production. *BMC Systems Biology*, 5(122), 2011. 21
- [138] Moreno P. *Bioinformatic methods for species-specific metabolome inference*. PhD thesis, University of Cambridge, 2012. 21, 117
- [139] Kotera M., Okuno Y., Hattori M., Goto S., and Kanehisa M. Computational assignment of the EC numbers for genomic-scale analysis of enzymatic reactions. *Journal of the American Chemical Society*, 126(50):16487–16498, 2004. 21
- [140] Rahman S. A. Metabolic pathway analysis web service (Pathway Hunter Tool at CUBIC). *Bioinformatics*, 21(7):

-
- 1189–1193, 2004. 21
- [141] Nakamura M., Hachiya T., Saito Y., Sato K., and Sakakibara Y. An efficient algorithm for *de novo* predictions of biochemical pathways between chemical compounds. *BMC Bioinformatics*, 13(Suppl 17):S8, 2012. 21
- [142] Matthew G., Swainston N., Dobson P., Jameson D., Simeonidis E., Smallbone K., and Malys N. Gap filling and identifying new reactions in metabolic networks based on metabolite similarity. *ICSB 2011*, 2011. 21
- [143] Henry C. S., Zinner J. F., Cohoon M. P., and Stevens R. L. *iBsu1103*: a new genome-scale metabolic model of *Bacillus subtilis* based on seed annotations. *Genome Biol*, 10(6), 2009. 21, 69, 93
- [144] Ott M. A. and Vriend G. Correcting ligands, metabolites, and pathways. *BMC Bioinformatics*, 7(517), 2006. 23, 203
- [145] Haraldsdóttir H. *Estimation of Transformed Reaction Gibbs Energy for Thermodynamically Constraining Metabolic Reaction Networks*. PhD thesis, Faculty of Industrial Engineering, Mechanical Engineering and Computer Science, University of Iceland, 2014. URL http://skemman.is/stream/get/1946/17308/40388/1/Hulda_S_Haraldsdottir_PhD_Thesis_2014.pdf. 23
- [146] May J. W., James A. G., and Steinbeck C. Metingear: a development environment for annotating genome-scale metabolic models. *Bioinformatics*, 29(17):2213–2215, 2013. 24, 115
- [147] May J. W. and Steinbeck C. Efficient ring perception for the Chemistry Development Kit. *Journal of Cheminformatics*, 6(3), 2014. 24
- [148] van Iersel M. P., Pico A. R., Kelder T., Gao J., Ho I., Hanspers K., Conklin B. R., and Evelo C. T. The BridgeDb framework: standardized access to gene, protein and metabolite identifier mapping services. 11(5), 2010. 29
- [149] Chambers J., Davies M., Gaulton A., Hersey A., Velankar S., Petryszak R., Hastings J., Bellis L., McGlinchey S., and Overington J. P. UniChem: A unified chemical structure cross-referencing and identifier tracking system. *Journal of Cheminformatics*, 5(3), 2013. 29
- [150] Krause F. semanticSBML: a tool for creating, checking, annotation and merging of SBML documents. Master’s thesis, Free University of Berlin, 2008. 30
- [151] Schulz M., Klipp E., and Liebermeister W. Propagating semantic information in biochemical network models. *BMC Bioinformatics*, 13(18), 2012. 30, 203
- [152] Mednis M., Brusbardis V., and Galvanauskas V. Comparison of genome-scale reconstructions using ModeRator. In *Proceedings of 13th IEEE International Symposium on Computational Intelligence and Informatics*, pages 79–84, 2012. 30
- [153] Sauls J. and Buescher J. Assimilating genome-scale metabolic reconstructions with modelBorgifier. *Bioinformatics*, 30(7):1036–1038, 2014. 30
- [154] Ullmann J. R. An algorithm for subgraph isomorphism. *Journal of the Association for Computing Machinery*, 23(1):31–42, 1976. 31
- [155] Cordella L. P., Foggia P., Sansone C., and Vento M. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(10):1367–1372, 2004. 31, 43
- [156] Morgan H. The generation of a unique machine description for chemical structures – a technique developed at Chemical Abstracts Service. *Journal of Chemical Documentation*, 5(2):107–113, 1965. 31
- [157] Wipke W. and Dyott T. Stereochemically unique naming algorithm. *Journal of the American Chemical Society*, 96(15):4834–4842, 1974. 31
- [158] Wipke W., Krishnan S., and Ouchi G. Hash functions for rapid storage and retrieval of chemical structures. *Journal of chemical information and computer sciences*, 18(1):32–37, 1978. 31
- [159] McKay B. D. Practical graph isomorphism. *Congressus Numerantium*, 30:223–232, 1980. 32
- [160] McKay B. D. and Piperno A. Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60:94–112, 2014. 32
- [161] Ihlenfeldt W. and Gasteiger J. Hash codes for the identification and classification of molecular structure elements. *Journal of Computational Chemistry*, 15(8):792–813, 1994. 32, 35
- [162] Agarwal K. and Gelernter H. A computer-oriented linear canonical notational system for the representation of organic structures with stereochemistry. *Journal of chemical information and computer sciences*, 34(3):463–479, 1994. 32, 197
- [163] Stephen E. Stein D. V. T., Stephen R. Heller. *InChI Technical Manual*. IUPAC, 2006. 33, 43, 148, 158, 168
- [164] Accelrys. Whitepaper: exact match searching. online, 2012. URL <http://accelrys.com/products/pdf/exact-match-searching.pdf>. accessed May 2014. 33
- [165] Marsaglia G. Xorshift RNGs. *Journal of Statistical Software*, 8(14):1–6, 2003. 35
- [166] Petrarca A., Lynch M., and Rush J. A method for generating unique computer structural representations of stereoisomers. *Journal of Chemical Documentation*, 7(3):154–165, 1967. 37, 186
- [167] Reed J. L., Vo T. D., Schilling C. H., and Palsson B. An expanded genome-scale model of *Escherichia coli* K-12
-

- (iJR904 GSM/GPR). *Genome Biol*, 4(9), 2003. 46, 101, 110
- [168] Oh Y.-K., Palsson B., Park S. M., Schilling C. H., and Mahadevan R. Genome-scale reconstruction of metabolic network in *Bacillus subtilis* based on high-throughput phenotyping and gene essentiality data. *J Biol Chem*, 282(39):28791–28799, 2007. 69
- [169] Drager A., Rodriguez N., Dumousseau M., Dorr A., Wrzodek C., Novère N. L., Zell A., and Hucka M. JSBML: a flexible java library for working with SBML. *Bioinformatics*, 27(15):2167–2168, 2011. 83, 84
- [170] ChemAxon. Calculator plugins, protonation. online. URL <http://www.chemaxon.com/marvin/help/calculations/protonation.html>. accessed May 2014. 84, 116
- [171] Liebermeister W., Krause F., Uhlenendorf J., Lubitz T., and Klipp E. semanticSBML: a tool for annotating, checking, and merging of biochemical models in SBML format. *Nature Precedings*, 2009. 84
- [172] Lister A. L., Pocock M., Taschuk M., and Wipat A. Saint: a lightweight integration environment for model annotation. *Bioinformatics*, 25(22):3026–3027, 2009. 84
- [173] Gille C., Hubner K., Hoppe A., and Holzhutter H.-G. METANNOGEN: annotation of biological reaction networks. *Bioinformatics*, 27(19):2763–2764, 2011. 85
- [174] Kay A. Dr. Alan Kay on the meaning of “object-oriented programming”. online. URL http://userpage.fu-berlin.de/~ram/pub/pub_jf47ht81Ht/doc_kay_oop_en. accessed May 2014. 86
- [175] Meyer F., Goesmann A., McHardy A. C., Bartels D., Bekel T., Clausen J., Kalinowski J., Linke B., Rupp O., Giegerich R., and Pühler A. GenDB – an open source genome annotation system for prokaryote genomes. *Nucleic Acids Research*, 31(8):2187–95, 2003. 87
- [176] Prlic A., Yates A., Bliven S. E., Rose P. W., Jacobsen J., Troshin P. V., Chapman M., Gao J., Koh C. H., Foisy S., Holland R., Rimsa G., Heuer M. L., Brandstatter-Muller H., Bourne P. E., and Willis S. BioJava: an open-source framework for bioinformatics in 2012. *Bioinformatics*, 28(20):2693–2695, 2012. 89
- [177] SBML Developers. libSBML API documentation. online. URL [http://sbml.org/Software/libSBML/docs/java-api/org.sbml.libsbml.Species.html#getCharge\(\)](http://sbml.org/Software/libSBML/docs/java-api/org.sbml.libsbml.Species.html#getCharge()). accessed May 2014. 92
- [178] Gonzalez O., Gronau S., Falb M., Pfeiffer F., Mendoza E., Zimmer R., and Oesterheld D. Reconstruction, modeling & analysis of *Halobacterium salinarum* R-1 metabolism. *Molecular BioSystems*, 4(2):148–159, 2008. 94
- [179] Shannon P., Markiel A., Ozier O., Baliga N., Wang J., Ramage D., Amin N., Schwikowski B., and Ideker T. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research*, 13(11):2498–2504, 2003. 95
- [180] Amid C., Birney E., Bower L., Cerdeno-Tarraga A., Cheng Y., Cleland I., Faruque N., Gibson R., Goodgame N., Hunter C., Jang M., Leinonen R., Liu X., Oisel A., Pakseresht N., Plaister S., Radhakrishnan R., Reddy K., Riviere S., Rossello M., Senf A., Smirnov D., Hoopen P. T., Vaughan D., Vaughan R., Zalunin V., and Cochrane G. Major submissions tool developments at the European nucleotide archive. *Nucleic Acids Research*, 40(D1):D43–D47, 2012. 95
- [181] ORACLE. Creating extensible applications. URL <http://docs.oracle.com/javase/tutorial/ext/basics/spi.html>. 96
- [182] The Apache Lucene open-source search software. online. URL <https://lucene.apache.org/>. accessed May 2014. 97
- [183] HyperSQL. online. URL <http://hsqldb.org/>. accessed May 2014. 97
- [184] Ostrovsky I. Efficient auto-complete with a ternary search tree. online. URL <http://igoro.com/archive/efficient-auto-complete-with-a-ternary-search-tree/>. accessed May 2014. 101
- [185] UniProt Knowledgebase. Controlled vocabulary of species. online. URL <http://www.uniprot.org/docs/speclist>. accessed May 2014. 101
- [186] Thiele I., Hyduke D. R., Steeb B., Fankam G., Allen D. K., Bazzani S., Charusanti P., Chen F.-C., Fleming R. M., Hsiung C. A., Keersmaecker S. C. D., Liao Y.-C., Marchal K., Mo M. L., Özdemir E., Raghunathan A., Reed J. L., Shin S.-I., Sigurbjörnsdóttir S., Steinmann J., Sudarsan S., Swainston N., Thijs I. M., Zengler K., Palsson B., Adkins J. N., and Bumann D. A community effort towards a knowledge-base and mathematical model of the human pathogen *Salmonella typhimurium* LT2. *BMC Systems Biology*, 5(8), 2011. 106
- [187] Haraldsdóttir H., Thiele I., and Fleming R. Comparative evaluation of open source software for mapping between metabolite identifiers in metabolic network reconstructions: application to Recon 2. *Journal of Cheminformatics*, 6(2), 2014. 107
- [188] Levenshtein V. Binary codes capable of correcting deletions, insertions and reversals. 10:707, 1966. 107
- [189] Krause S., Willighagen E., and Steinbeck C. JChemPaint – using the collaborative forces of the internet to develop a free editor for 2D chemical structures. *Molecules*, 5:93–98, 2000. 111
- [190] Sayle R. and Delany J. Canonicalization and enumeration of tautomers. online, 1999. accessed May 2014. 112, 160
- [191] Novère N. L. SBML Level 3 package, annotations. online, 2012. URL <http://sbml.org/Documents/>

-
- [Specifications/SBML_Level_3/Packages/Annotations_%28annot%29](#). accessed May 2014. 116
- [192] Settimo L., Bellman K., and Knegtel R. Comparison of the accuracy of experimental and predicted pK_a values of basic and acidic compounds. *Pharm Res*, 31:1082–1095, 2014. 116
- [193] Sayle R. Physiological ionization and pK_a prediction. online, 2000. URL <http://www.daylight.com/meetings/emug00/Sayle/pkapredict.html>. accessed May 2014. 116
- [194] Flamholz A., Noor E., Bar-Even A., and Milo R. eQuilibrator – the biochemical thermodynamics calculator. *Nucleic Acids Research*, 40(D1):D770–D775, 2012. 116
- [195] Razinger M., Balasubramanian K., Perdih M., and Munk M. Stereoisomer generation in computer-enhanced structure elucidation. *J. Chem. Inf. Comput. Sci.*, 33(6):812–825, 1993. 122, 181
- [196] Berger F., Flamm C., Gleiss P., Leydold J., and Stadler P. Counterexamples in chemical ring perception. *Journal of chemical information and computer sciences*, 44(2):323–331, 2004. 122, 123
- [197] Downs G., Gillet V., Holliday J., and Lynch M. Review of ring perception algorithms for chemical graphs. *Journal of chemical information and computer sciences*, 29(3):172–187, 1989. 122
- [198] Downs G. Ring perception. In Gasteiger J., editor, *Handbook of Chemoinformatics*, volume 1, chapter 5.2, pages 166–177. Wiley VCH, 2003. 122, 124, 125, 144
- [199] Sedgewick R. and Wayne K. Graphs. In *Algorithms*, chapter 4, pages 524–527. 4th edition, 2011. 125, 129
- [200] Hanser T., Jauffret P., and Kaufmann G. A new algorithm for exhaustive ring perception in a molecular graph. *Journal of Chemical Information and Computer Sciences*, 36(6):1146–1152, 1996. 126, 134
- [201] Nikolova-Jeliazkova N. Slow fingerprints? *CDK News*, 2(2):34–40, 2005. 126, 240
- [202] Kruskal J. B. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956. 126, 131, 240
- [203] Berger F., Gritzmam P., and Vries S. Minimum cycle bases for network graphs. *Algorithmica*, 40:51–62, 2004. 126
- [204] Skiena S. Graph Problems: Polynomial-Time, Minimum Spanning Tree. In *The Algorithm Design Manual*, chapter 15, pages 484–489. Springer, New York, 2nd edition, 2008. 131
- [205] Hopcroft J. and Tarjan R. Efficient algorithms for graph manipulation. *Communications of the ACM*, 16(6):372–378, 1973. 131
- [206] Vismara P. Union of all the minimum cycle bases of a graph. *Electr. J. Comb.*, 4:73–87, #R9, 1997. 132, 133
- [207] Horton J. A polynomial-time algorithm to find the shortest cycle basis of a graph. *SIAM Journal on Computing*, 16(2):358–366, 1987. 133
- [208] National Cancer Institute. National Cancer Institute (NCI) database download. online. URL <http://cactus.nci.nih.gov/download/nci/>. accessed May 2014. 135
- [209] Irwin J., Sterling T., Mysinger M., Bolstad E., and Coleman R. ZINC: a free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 52(7):1757–1768, 2012. 135
- [210] Gaulton A., Bellis L. J., Bento A. P., Chambers J., Davies M., Hersey A., Light Y., McGlinchey S., Michalovich D., Al-Lazikani B., and Overington J. ChEMBL: a large-scale bioactivity database for drug discovery. *Nucleic Acids Research*, 40(D1):D1100–D1107, 2012. 135
- [211] Schwerdtfeger P., Wirz L., and Avery J. Program Fullerene: A software package for constructing and analyzing structures of regular fullerenes. *Journal of Computational Chemistry*, 34(17):1508–1526, 2013. 137
- [212] Kolodzik A., Urbaczek S., and Rarey M. Unique Ring Families: A chemically meaningful description of molecular ring topologies. *Journal of chemical information and modeling*, 52(8):2013–2021, 2012. 143
- [213] Rybalkin M. The Blue Oblisk Exchange: Aromaticity perception differences. URL <http://blueobelisk.shapado.com/questions/aromaticity-perception-differences>. 148
- [214] Hückel E. Quantentheoretische beiträge zum benzolproblem I. Die elektronenkonfiguration des benzols und verwandter verbindungen. *Z. Phys.*, 70(3-4):204–208, 1931. 148
- [215] Hückel E. Quantentheoretische beiträge zum benzolproblem II. Quantentheorie der induzierten polaritäten. *Z. Phys.*, 76(5-6):310–337, 1931. 148
- [216] Balaban A. T. Applications of graph theory in chemistry. *Journal of chemical information and computer sciences*, 25(3):334–343, 1985. 148
- [217] Mockus J. and Stobaugh R. The Chemical Abstracts Service chemical registry system. VII. Tautomerism and alternating bonds. *Journal of Chemical Information and Computer Sciences*, 20(1):18–22, 1980. 151
- [218] Sayle R. A. So you think you understand tautomerism? *J Comput Aided Mol Des*, 24(6-7):485–496, 2010. 151, 171
- [219] Sitzmann M., Ihlenfeldt W.-D., and Nicklaus M. C. Tautomerism in large databases. *J Comput Aided Mol Des*, 24(6-7):521–551, 2010. 151
- [220] Warr W. A. Tautomerism in chemical information management systems. *J Comput Aided Mol Des*, 24(6-7):497–520, 2010. 151, 157
-

-
- [221] Thalheim T., Vollmer A., Ebert R.-U., Kühne R., and Schüürmann G. Tautomer identification and tautomer structure generation based on the InChI code. *Journal of Chemical Information and Modeling*, 50(7):1223–1232, 2010. 151
- [222] Kochev N. T., Paskaleva V. H., and Jeliaskova N. Ambit-Tautomer: An open source tool for tautomer generation. *Mol. Inf.*, 32(5-6):481–504, 2013. 151
- [223] Sayle R. Cheminformatics toolkits: a personal perspective. online, 2012. accessed May 2014. 153, 166
- [224] Hopcroft J. E. and Karp R. M. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973. 155
- [225] Edmonds J. Paths, trees, and flowers. *Canad. J. Math*, 17:449–467, 1965. 155
- [226] Micali S. and Vazirani V. An $O(|E|\sqrt{|V|})$ algorithm for finding maximum matching in general graphs. In *SFCS '80 Proceedings of the 21st Annual Symposium on Foundations of Computer Science*, pages 17–27, 1980. 157
- [227] Vazirani V. V. A simplification of the MV matching algorithm and its proof. arXiv:1210.4594, 2013. 157
- [228] Hansen P. and Zheng M. Assigning a Kekulé structure to a conjugated molecule. *Computers Chem.*, 19(1):21–26, 1994. 157, 159
- [229] Daylight CIS. DEPICT: Interactive depiction of SMILES. online. URL <http://www.daylight.com/daycgi/depict>. accessed May 2014. 163
- [230] James C. OpenSMILES specification. online, 2007. URL <http://www.opensmiles.org/opensmiles.html>. accessed May 2014. 164, 166
- [231] O’Boyle N. Towards a Universal SMILES representation—a standard method to generate canonical SMILES based on the InChI. *Journal of Cheminformatics*, 4(22), 2012. 166
- [232] Neglur G., Grossman R., and Liu B. Assigning unique keys to chemical compounds for data integration: Some interesting counter examples. In Ludäscher B. and Raschid L., editors, *Data Integration in the Life Sciences*, volume 3615 of *Lecture Notes in Computer Science*, pages 145–157. Springer Berlin Heidelberg, 2005. 166
- [233] Urbaczek S., Kolodzik A., and Rarey M. The Valence State Combination Model: A generic framework for handling tautomers and protonation states. *Journal of chemical information*, 54(3):756–766, 2014. 169
- [234] Cieplak T. and Wisiewski J. L. A new effective algorithm for the unambiguous identification of the stereochemical characteristics of compounds during their registration in databases. *Molecules*, 6(11):915–925, 2001. 178
- [235] O’Boyle N. M. Name that stereochemistry - when mol files go wrong. online. URL <http://baoilleach.blogspot.co.uk/2010/12/name-that-stereochemistry-when-mol.html>. accessed May 2014. 180
- [236] Brecher J. Graphical representation of stereochemical configuration (IUPAC recommendations 2006). *Pure Appl. Chem.*, 78(10):1897–1970, 2006. 181
- [237] Carbonell P., Carlsson L., and Faulon J. Stereo signature molecular descriptor. *Journal of chemical information and modeling*, 53(4):887–897, 2013. 185
- [238] Rahman S., Bashton M., Holliday G. L., Schrader R., and Thornton J. M. Small Molecule Subgraph Detector (SMSD) toolkit. *Journal of Cheminformatics*, 1(12), 2009. 187, 241
- [239] IUPAC. Specification of configuration and conformation (provisional recommendations). In *Preferred IUPAC Names*, chapter 9. 2004. 187, 188
- [240] Prelog V. and Helmchen G. Basic principles of the CIP-system and proposals for a revision. *Angew. Chem. Int. Ed. Engl.*, 21:567–583, 1982. 187, 188
- [241] Willighagen E. CIP rules for stereochemistry. online, 2010. URL <http://chem-bla-ics.blogspot.nl/2010/04/cip-rules-for-stereochemistry.html>. accessed May 2014. 188
- [242] Labute P. An efficient algorithm for the determination of topological RS chirality. *Journal of the Chemical Computing Group*, 1996. 193
- [243] Batchelor C., Karapetyan K., Tkachenko V., and Williams A. Validation and standardization of molecular structures in general and sugars in particular: a case study. In *6th Joint Sheffield Conference on Chemoinformatics*, 2013. 197
- [244] Torrance G. Using the CDK’s group module. online, 2012. URL <http://gilleain.blogspot.co.uk/2012/11/using-cdks-group-module.html>. accessed May 2014. 197
- [245] Kiss R. Double bond stereochemistry? not yet solved... online. URL <http://blog.mcule.com/2011/09/double-bond-stereochemistry-not-yet.html>. accessed May 2014. 197
- [246] Bellis L. To remove or not to remove – that is the question. online, 2013. URL <http://chembl.blogspot.co.uk/2013/03/to-remove-or-not-to-remove-that-is.html>. accessed May 2014. 198
- [247] Kharchenko P., Chen L., Freund Y., Vitkup D., and Church G. Identifying metabolic enzymes with multiple types of association evidence. *BMC Bioinformatics*, 7(177), 2006. 203
- [248] Zhang T., Li H., Xi H., Stanton R., and Rotstein S. HELM: A hierarchical notation language for complex

-
- biomolecule structure representation. *Journal of chemical information and modeling*, 52(10):2796–2806, 2012. 205
- [249] Smallbone K. S. K. Striking a balance with Recon 2.1. arXiv:0902.0885, 2013. 205
- [250] May J. CDK now built using Maven. online, 2014. URL <http://efficientbits.blogspot.co.uk/2014/02/cdk-now-built-using-maven.html>. accessed May 2014. 205
- [251] Rahman S. and Schomburg D. Observing local and global properties of metabolic pathways: ‘load points’ and ‘choke points’ in the metabolic networks. *Bioinformatics*, 22(14):1767–1774, 2006. 236
- [252] Moszer I., Glaser P., and Danchin A. SubtiList: a relational database for the *Bacillus subtilis* genome. *Microbiology*, 141:261–268, 1995. 237
- [253] Altschul S. F., Gish W., Miller W., Myers E. W., and Lipman D. J. Basic Local Alignment Search Tool. *J Mol Biol*, 215(3):403–10, 1990. 237