FCT
Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA EDUCAÇÃO E CIÊNCIA

POPH
QUALIFICAR É CRESCER.

QREN
QUADRO
DE REFERÊNCIA
ESTRATÉGICO
NACIONAL
PORTUGAL2007.2013

Governo da
República Portuguesa

UNIÃO EUROPEIA
Fundo Social Europeu

Filipe Alexandre Wang Liu

Evaluation and Development of Algorithms
and Computational Tools for Metabolic Pathway Optimization

UMinho|2018

Universidade do Minho
Escola de Engenharia

Filipe Alexandre Wang Liu

Evaluation and Development of Algorithms and Computational Tools for Metabolic Pathway Optimization

The MAP Doctoral Program in Computer Science of the Universities Minho, Aveiro and Porto

Universidade do Minho

universidade de aveiro

U.PORTO

March 2018

Universidade do Minho
Escola de Engenharia

Filipe Alexandre Wang Liu

Evaluation and Development of Algorithms and Computational Tools for Metabolic Pathway Optimization

The MAP Doctoral Program in Computer Science
of the Universities Minho, Aveiro and Porto

Universidade do Minho

universidade de aveiro

U.PORTO

This work was executed under the supervision of:
Professor Miguel Francisco de A. Pereira da Rocha
and
Professor Isabel Cristina de A. Pereira da Rocha

March 2018

DECLARAÇÃO

Nome: Filipe Alexandre Wang Liu

Endereço electrónico: filipeliu1@gmail.com

Número do Bilhete de Identidade: 15946533

Título da tese: Evaluation and Development of Algorithms and Computational Tools for Metabolic Pathway Optimization

Orientador(es):

Professor Doutor Miguel Francisco de Almeida Pereira da Rocha

Professora Doutora Isabel Cristina de Almeida Pereira da Rocha

Ano de conclusão: 2018

Designação do Doutoramento:

Informática - MAP-i

Universidade do Minho, 29/03/2018

Assinatura: [illegible]

STATEMENT OF INTEGRITY

I hereby declare having conducted my thesis with integrity. I confirm that I have not used plagiarism or any form of falsification of the results in the process of the thesis elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

University of Minho, March 29th, 2018

[illegible]

Filipe Alexandre Wang Liu

Acknowledgements

It's been a long journey with so many people involved. First of all, I would like to express a huge gratitude to my supervisor Prof. Miguel Rocha, for all the support and stamina to handle my stiff mood, without him this thesis would have never existed, neither my academic career.

I would like to thank Prof. Isabel Rocha for co-supervising my thesis, and who contributed greatly for the scientific output of this work.

To all members (and ex-embers) of the BiSBII group, for all the great moments, the retreats, the cakes. A special thanks to Sophia, who collaborated directly with my work, but also for tutoring me with "wet lab tours". To Joana and Fábio that provided me tremendous input regard to SBML models. To Oscar for been an annoying and good friend.

To all people at the Henry Lab in Argonne for receiving me, I learnt a lot during these 6 months. A special thanks to José, for bringing me to Argonne, but also for all the help and friendship during my stay in Chicago.

To Bruno for the coffee moments and weekend lunch's, we had great discussions about HPC, politics and the universe.

I would like to thank the Portuguese Foundation for Science and Technology for the PhD grant SFRH/BD/111490/2015.

Finally, I would like to thank my parents for their unconditional support.

Braga, March 2018

Filipe

Abstract

Metabolic engineering exploits microorganisms to build cell factories, allowing to produce valuable compounds from their enzymatic machinery. It involves the selection of an organism, along with a set of genetic modifications to optimize the process. Information regarding biological mechanisms are scattered among the literature. Metabolic databases provide a centralized platform compiling existing biological data to build a catalog of all known enzymatic transformations across all domains of life.

The development of genome-scale metabolic models allows to expose all possible biochemical transformations that an organism can offer. Computer algorithms use these models to exploit the capabilities and limitations of the organisms. Constraint-based modeling approaches allow to predict phenotype given modifications in the network. In recent years, there has been a significant increase in the number of available models, and for certain organisms several models were built. The accuracy of these methods is in many cases dependent on the quality of these models, that is limited to the available information in the literature (or databases).

This thesis improves the existing methods by developing better data management strategies for the metabolic modeling community. Metabolic databases are usually the input data for many modeling tools, and the quality of solutions depends on the quality of the databases. Currently, several metabolic databases exist, most of them sharing a common set of information, and there is a need for a centralized system to take the most advantage of their content. However, each database adopts its own naming system to catalog its instances, being in many cases, difficult to compare with others.

An integration pipeline is here designed to fuse metabolic databases into a common namespace allowing better analysis of the entire metabolic catalog across several databases, and exploring different methods to reconcile the metabolites and reactions included in these databases.

In a second part of this work, the Systems Biology Markup Language which is the most common medium to store and represent genome-scale metabolic models is analyzed. Like databases, models also adopt unique nomenclatures for reactions and compounds. Here, methods to annotate metabolites and reactions in models are developed allowing to connect models with database instances, thus allowing to adopt a single naming system for their entities. The purpose of the methods is to standardize the entire model, therefore, other entities such as, genes, compartments, simulation media, are also considered to unify these models. The standardization methods were implemented in the KBase platform, which allows to improve the compatibility of this system with models built from external tools.

In the last part of this thesis, the pathway enumeration problem is revisited. Synthetic biology explores cellular modifications to produce valuable products by inserting enzymatic capabilities of other organisms. The selection of suitable set of genes is highly combinatorial, since in many cases there are several alternatives to reach the target product. A common limitation of most of the existing methods is the inability to fully explore this combinatorial space. In this work, the (hyper)graph methods are analyzed and improved to fully enumerate biological pathways. As result, two existing algorithms were improved regarding to scalability, allowing to fully enumerate larger solution sets.

Resumo

Um dos objetivos da Engenharia Metabólica é a síntese de compostos de valor acrescentado através de microrganismos. Uma das etapas deste processo envolve a seleção de organismos em combinação com alterações genéticas que permitem otimizar este processo. As bases de dados metabólicas centralizam os dados biológicos disponibilizando um catálogo de todo o conhecimento existente relacionado ao contexto enzimático.

A reconstrução de modelos metabólicos à escala genómica permite estudar os processos metabólicos dos diversos organismos. Com o recurso a métodos computacionais, estes modelos permitem expor as capacidades e limitações dos diversos organismos. Abordagens como a modelação baseada em restrições permitem prever fenótipos dadas alterações nas vias metabólicas. Nas últimas décadas, houve um aumento significativo do número de modelos publicados, e para alguns organismos existem várias versões disponíveis. A capacidade de previsão destes modelos está dependente da informação disponível nas bases de dados e na literatura.

Esta tese visa melhorar os métodos anteriores abordando questões relacionadas com a integração de dados. As bases de dados metabólicas são geralmente a principal fonte de informação para os métodos existentes, implicando diretamente na capacidade de resolução destes problemas. Atualmente, existem várias bases de dados biológicas, havendo uma necessidade de desenvolver sistemas centralizados. No entanto, é comum estes adotaram identificares próprios, não sendo possível executar uma comparação direta. Neste trabalho, foram desenvolvidas estratégias para reconciliar bases de dados no contexto metabólico, permitindo integrar compostos e reações.

Na segunda parte deste trabalho, este processo de integração foi expandido para incluir modelos metabólicos à escala genómica. De forma semelhante às bases de dados, os modelos adotam também identificadores próprios para representar compostos e reações. Para unificar modelos, foram desenvolvidos métodos de anotação que permitem relacionar as instâncias dos modelos com as bases de dados. Foram, também, implementadas estratégias para identificar genes, compartimentos e as restrições da simulação. Neste trabalho, os métodos forma implementados na plataforma KBase, permitindo melhorar a compatibilidade do sistema com os modelos externos.

Por fim, vários métodos de enumeração de vias metabólicas foram abordados. A biologia sintética visa manipular o metabolismo celular para produção de compostos através da inserção de genes. A seleção destes genes é um problema combinatório, que, dado um composto alvo, identifica vários conjuntos de genes capazes de concretizar a via sintética. Neste trabalho, pretende-se melhorar a capacidade de enumerar todas as vias possíveis, dado um conjunto limitado de reações e o tamanho das vias. Como resultado, foram melhorados dois métodos existentes baseados em hipergrafos, melhorando a escalabilidade destes métodos permitindo enumerar problemas ou vias de maior dimensão.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context and Motivation

The re-engineering of cells for *de novo* synthesis of desirable metabolites involves several steps, from data collection and curation, to optimal strain selection, pathway identification and analysis of the best solutions.

Research in computational systems biology develops software tools to predict phenotype responses given environmental and genetic modifications. In the past years, a vast catalog of Bioinformatics software was developed to fit many topics in this field [21], ranging from network reconstruction and representation problems to data visualization and metabolic network analysis. Despite this effort, most topics still present a big challenge to software development, since the reconstruction, analysis and optimization of large scale metabolic networks still face many challenges.

Building computer models to predict cellular behavior is extremely data intensive. For many organisms, the biological mechanisms are inferred from similar species to fill the knowledge gaps. This makes data integration and analysis an important task to build more accurate models.

Biological databases compile data from literature and biochemical knowledge, and they are essential tools for many bioinformatics methods. Each database specializes into a specific topic and there is an increasing need to implement methods that allow combining and integrating distinct sources of data [44].

Because of the massive amount of information and expertise required to model biological systems, several community collaborations emerged to *curate* the representation of many levels of omics data [13]. As an example, the Gene Ontology (GO) was created to design a standard method to represent cellular mechanisms and functions that in most cases were defined as textual expressions.

In the software perspective, in the past years, many tools were developed to perform important tasks related to this research topic. Metabolic Engineering is powered by *in silico* analysis and, therefore, there is a demand for specialized integrated development environments, that still offer a challenge for software engineers [57]. Libraries dedicated to genome-scale modelling take advantage of the scripting environment of existing technologies to provide essential methods for analysis. The COnstraint-Based Reconstruction and Analysis methods [111] COBRA and COBRApy [32] are libraries for MatLab and Python, which provide many essential methods and tools to conduct constraint-based analysis studies. Standalone software programs such as OptFlux [107] integrate many techniques to tackle problems in this field, reducing the learning curve to apply these methods for an audience with less programming skills.

Advanced computational tools are currently able to identify optimal pathways through stoichiometric network analysis by either computing algebraically steady states of the stoichiometric network or from graph topological analysis. The computation of steady states relies on the analysis of the feasible solutions which represent all possible steady state flux distributions. Extreme Pathways (EP) [8] and Elementary Flux Modes (EFM) [114] both compute flux vectors through convex analysis [33].

However, the interoperability of these platforms is essential since no platform is capable to implement all existing methods. The Systems Biology Markup Language [58] was created to provide a common medium to share cellular models. Because of the flexibility of the format to cover a variety of fields in systems biology, but also due to the limitations of the earlier versions of the format, many tools implemented their own methods to represent several aspects regarding genome-scale modeling [31, 105].

Several tools were developed for model reconstruction, each adopting a variety of strategies to assemble genome-scale models. Today there are several hundreds of curated models, and if accounting for automated generated models these numbers go up to many thousands [89, 84]. Because of compound and reaction aliasing, it is difficult to directly compare models generated from different tools. It is also difficult to correctly interpret the mathematical components of the model because of the lack of proper standards [19, 31]. These problems diminish the re-usability of existing models for future research.

1.2 Objectives

This thesis is dedicated to improve metabolic modeling and pathway optimization approaches by defining better strategies to manage and integrate their data context.

More specifically, the work will address the following scientific/ technological goals:

1. Integration of metabolic data sources:

    Access existing biochemical databases;

    Develop unification strategies to generate an unified database;

    Implement a pipeline to integrate metabolites and reactions.

2. Standardization of genome-scale metabolic models:

    Explore existing methods to represent GSM components in SBML;

    Develop standardization strategies to conform genome-scale metabolic models.

    Develop a Software tool that is capable to standardize SBML models.

    Assess the proposed strategies with existing SBML models.

3. Enumeration synthetic pathways:

Explore, validate and extend optimization algorithms that allow searching over metabolic networks for the best routes from sets of source metabolites to target metabolites.

Validate the solution with a case study.

1.3 Structure of the Thesis

This thesis is divided into five chapters, being their contents described bellow.

In this first chapter, *Introduction*, we provided a brief introduction of the motivation and the main goals of this work.

The second chapter, *Metabolic Database Integration*, introduces several important aspects related to databases that are relevant to the metabolic modeling community. The chapter is dedicated to improve existing methods to unify these databases, by exploring better approaches and implement a standard pipeline to generate consensus databases.

The third chapter, *Standardization of Genome-Scale Models*, tackles the genome-scale model representation by focusing the Systems Biology Markup Language modeling strategies. This chapter explores existing variations to represent these models, and propose standardization methods to reshape genome-scale metabolic models.

The fourth chapter, *Pathway Optimization*, explores synthetic pathway optimization methods. This chapter is dedicated to explore existing alternatives of pathway enumeration.

Lastly, the fifth chapter, *Conclusions*, presents the main conclusions of the work, also proposing future research topics.

1.4 Scientific Output

Publications :

Liu, F., Vilaça, P., Rocha, I., Rocha, M. (2015). Development and application of efficient pathway enumeration algorithms for metabolic engineering applications. Computer

Methods and Programs in Biomedicine, 118(2), 134–146.

Liu, F., Rocha, I., Rocha, M. (manuscript in preparation). Automated generation of integrated metabolic databases.

Liu, F., Faria, J., Henry, C, Rocha, I., Rocha, M. (manuscript in preparation). SBML-Tools: a KBase module to annotate and standardize SBML models.

(equal contribution) Santos, S., **Liu, F.**, Costa, C., Vilaça, P., Rocha, M., Rocha, I. (manuscript in preparation). MIYeasTK: The Metabolic Integrated Yeast Knowledgebase.

Poster Presentations :

Liu, F., Rocha, I., Rocha, M. (2017). Metabolic integration using graph databases. Great Lakes Bioinformatics Conference 2017. Chicago, Illinois, USA, May 15-17

Oral Presentations :

Liu, F., Xavier, J., Ramalho, F., Rocha, M., Rocha, I. (2017). Unification of genome scale models. Metabolic Pathway Analysis 2017. Bozeman, Montana, USA, July 24-28.

Other Collaborations :

(oral) Santos, S., **Liu, F.**, Costa, C., Vilaça, P., Rocha, M., Rocha, I. (2016). MIYeastK: The Metabolic Integrated Yeast Knowledgebase. SPB2016 - Book of Abstracts XIX National Congress of Biochemistry. No. O5/03, Guimarães, Portugal, Dec 8-10, 37

(poster) Xavier, J., **Liu, F.**, Ramalho, F., Rocha, I. (2017). Standardization, Completion and New Predictions of 121 Manually Curated Genome-Scale Prokaryotic Models. Copenhagen Biosciences Conferences - 11th Conference: Data-Driven Biotechnology, Bench, Bioreactor and Bedside - Abstract Book. Copenhagen, Denmark, 7-11 May

(oral) Faria, J., Edirisinghe J., **Liu, F.**, Henry C., (2017). Improving automated model reconstruction. Metabolic Pathway Analysis 2017. Bozeman, Montana, USA, July 24-28, 48

(oral) Edirisinghe J., Faria, J., **Liu, F.**, Xavier, J., Seaver, S., Weisenhorn, P., Jeffryes, J., Gu, T., Zhang, Q., Rocha, I., Henry, C. (2017). Automated pathway curation and improving metabolic model reconstruction based on phylogenetic analysis of pathway conservation. ICSB 2017 - 18th International Conference on Systems Biology. Virginia, USA, Aug 6-12

(submitted for conference) Edirisinghe J., Faria, J., **Liu, F.**, Henry, C., (2018). Automated Reconstruction and Comparison of Metabolic Models for Diverse Fungal Genomes. Metabolic Engineering 12, Munich, Germany, June 24-28.

(submitted for conference) Faria, J., Edirisinghe J., Seaver, S., **Liu, F.**, Weisenhorn, P., Jeffryes, J., Gu, T., Zhang, Q., Henry, C., (2018). Improving Automated Model Reconstruction across Phylogenetically Diverse Genome-Scale Metabolic Models. Metabolic Engineering 12, Munich, Germany, June 24-28.

(manuscript in preparation for journal article) Xavier, J., **Liu, F.**, Ramalho, F., Rocha, I. Standardization, Completion and New Predictions of 121 Manually Curated Genome-Scale Prokaryotic Models.

Chapter 2

Metabolic Database Integration

Abstract

Metabolic (pathway) databases are catalogs of all known enzymatic biochemistry found in the literature, describing enzymes in a functional approach rather than just a descriptive annotation and being used as building blocks for genome-scale metabolic reconstructions. Currently, there are several metabolic databases, along with many dedicated to catalog small molecules. To capitalize on this ecosystem of databases, it is desirable to have a unified catalog.

In this work, a comprehensive analysis of several of the most popular metabolic databases is conducted, to study their degree of heterogeneity and propose approaches to generate an unified catalog. An integration system is implemented that is capable of generating on-demand, and in a fully automated way, an integrated database of reactions and metabolites from several sources. A graph database is used to store the raw entities, which proven to facilitate the analysis of the integration of the data sources. The proposed methods for entity resolution allows users to configure the certainty level of the automated merges of duplicates in the data sources.

The pipeline is tested to generate an integrated database, and it is compared against

a manually curated set, but also with the integrated database MetaNetX to assess its performance. The unified database created by this work can be explored with a web application.

2.1 Introduction

Computer-aided design in the biological field is a highly data oriented exercise. Arguably, this is one of the leading causes of the growth of bioinformatics data every year. The increase of the computational power and the development of better omics methods and technologies led to a significant increase of the number of databases, and their size each year. The better support for crowd sourcing protocols also promoted the data boost in biological repositories.

Metabolic databases are no exception, and currently there are hundreds of databases for small molecules and biological pathways, increasing substantially in the last decades. This phenomenon was already defined as a ”loose federation of bio-nations” [43]. One of the reasons for the proliferation of biological resources is the inability for users to contribute or reshape existing data repositories, which in several situations forced researchers to develop their own alternative databases to fit their needs. A related reason is the need for specialized repositories for specific topics [64, 118] (e.g., a disease, a organism, a class of chemical compounds), where replicates of existing databases to reshape the logic for a specific topic may allow a better understanding of the data. Lastly, the publication of repositories is a method to gain reputation in a particular research field.

The increase of biological databases enriches the community, but may also pollute the existing data ecosystem. Multiple chain references between databases may cause error propagation, but also independent systems have different update cycles and maintenance frequencies, which may cause inconsistencies (e.g., incoherent information, dead references, etc). In general, this requires added human effort to keep all the databases updated and clean. Indeed, information management became a relevant topic in science, engineering, and biomedical fields [30].

A common problem of biological databases is the high redundancy between data. Each database adopts its own identifiers, making difficult to detect the unique instances, but also to merge or translate duplicate information between databases. The advantage of having a rich global data pool is diminished, because of the lack of interoperability between distinct resources, making the effort to unify systems in many cases not affordable or justified.

Existing integrated solutions for metabolic databases address this problem by offering access to a unified reference space of biochemical databases. Tools such as the Chemical Translation Service [125] or UniChem [17] offer a web service that allows to convert among different database identifiers or from those into molecular properties (e.g., names, structures, etc). The other approach is to materialize the unified space into a consensus database [9, 74]. In this case, a new database is generated from existing metabolic resources, providing an integrated repository.

The BKM-react [76] (now the BKMS-react) is a reconciliation that integrated the BRENDA with KEGG and MetaCyc (later with the addition of SABIO-RK). The integration is limited to reactions only.

The MetRxn database [74] provides an integrated resource of KEGG, MetaCyc, BREANDA and Reactome plus several genome scale models into a unified dataset. The database was built because of the need of an integrated biochemistry set for quality model reconstruction. It includes 90 genome-scale models along with the metabolic databases.

With an identical purpose of the MetRxn, the ModelSEED [54] was created for the purpose of genome-scale model reconstruction applications. It serves as a base biochemistry set for automated scaffold of metabolic reconstructions. It initially was based on the KEGG database merged with 13 published models, and later updated with the MetaCyc.

The MetaNetX/MNXRef [9, 90] is the largest integration currently available, integrating more than a hundred thousand compounds of a wide range of databases, including all of the previously mentioned. The first version of the database unified 11 databases (KEGG, MetaCyc, ChEBI, ModelSEED, etc), and several new resources were added since, while also some of the old discontinued databases were discarded.

Automated integration is subject to errors, and in many cases this is inevitable since these problems are caused by inconsistent information from the sources. No software tools are provided to rebuild these databases, and the responsibility to update these integrations is given to their owners.

This work explores metabolic database integration methods, being current implementations revised to understand the state of the art of data integration regarding the metabolic universe.

A survey of several popular databases of small molecules and biological pathways is conducted to assess their data content and their architecture for unification purposes. Alternative methods are proposed to design an integration pipeline capable of generating an integrated reference space of metabolic databases.

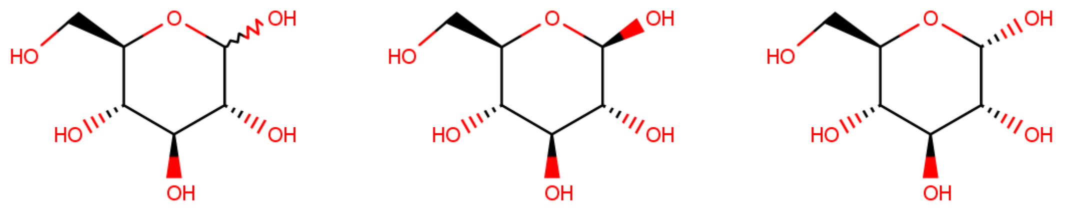
We propose several integration rules that include both positive and negative rules to allow fine tunning the integration approaches. Each of these rules targets metabolites that shares common properties (e.g., chemical formula, structure, name, etc) allowing to match for both positive and negative similarity. All rules are weighted, allowing users to customize their impact in the integration.

To assemble an integrated unified metabolic database, a framework is developed to conduct all the necessary steps from data extraction up to entity resolution methods. The framework is used to generate an integrated database of KEGG, MetaCyc and BiGG, and it is compared against the latest reference space of the MetaNetX integrated database. To benchmark the result an extensive manually curated dataset is used to compare with the solution provided by this work and MetaNetX. The generated approach obtained a decrease of 50% on false positives, with a cost of increasing 11% the number of false negatives.

The pipeline is distributed in a Docker container, with all required source code and dependencies [1], allowing users to assemble integrated databases, but also programmatically extensible to other data sources that were not included in this study.

A dedicated web application was created to explore the integrated solution, it provides a standalone database (https://fxe.github.io/biodb/) for publishing and curation.

[1]GitHub repository: https://github.com/Fxe/biosynth-framework



(a) D-Glucose (b) alpha-D-Glucose (c) beta-D-Glucose

Figure 2.1: Glucose structures. The connectivity layer includes atoms and bond valence, the stereo layer adds additional spatial arrangement in a 3D space.

2.2 Molecules, Biochemistry and Databases

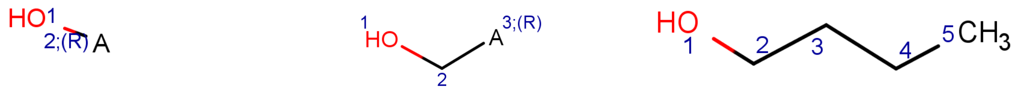
2.2.1 Representing Compounds

In a metabolic engineering perspective, in many publications, a compound, molecule or metabolite have the same meaning, since they all refer to small molecules, most of them organic since they are participants of cellular metabolism. Thus, they will be treated as a single concept in the following text and in this work. A chemical molecule, as a computer representation, is a simple undirected graph where vertices are atoms and edges are the chemical bonds (Figure 2.1a). Although, for both vertices and edges they must be labeled to define the atom type and the bond type.

However, not all molecules are flat (Figure 2.1), being the geometry of a compound defined by its stereochemistry. In the figure, solid or dashed wedge are used to represent atoms below or above the plane.

Also not all molecules are fully defined in computational representations. The R-groups or "Markush" structures (Figure 2.2a) are undefined groups to represent abstract structures. The use of R-groups to define incomplete molecules allows to create molecular hierarchies to group compounds into categories (Figure 2.2). As an example, the generic alcohol (Figure 2.2a) is a substructure for alcohol molecules (i.e., ethanol, methanol, propanol, butanol, etc), but also a substructure for many other structures where it fits.

In some cases, the R-group is a generic placeholder for a family of molecules. The DNA



(a) An Alcohol. (b) A short-chain primary alcohol. (c) Butanol.

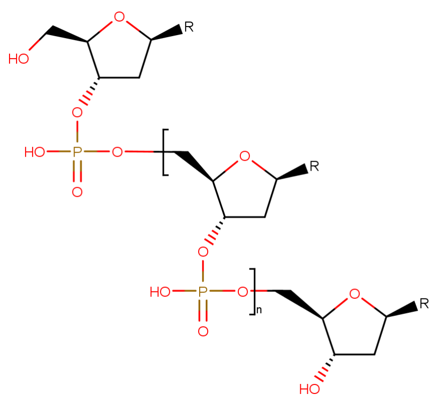
Figure 2.2: R-groups may define subclass of molecules. SMILES representation: a) - O[R], b) - OC[R], c) - OCCCC. a), b), c) are sub instances with the following substitutions: a) $\xrightarrow{[R]\to C[R]}$ b), b) $\xrightarrow{[R]\to CCC}$ c)

is an example (Figure 2.3a) that contains placeholders for cytosine, thymine, adenine and guanine molecules in its chains.

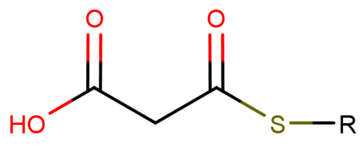
In general, the R-group allows to define a variable part of the structure. But not all R-groups are used to define a small molecule. The acyl carrier protein is an important component in the fatty acid biosynthesis, in its attached to a substrate form, the acyl carrier protein is represented by the R-group (Figure 2.3b) since its structure is non definable, but in this case the R-group is specific to the protein.

Polymers are represented by a repeating unit, that encloses a structure region marked as variable, allowing to define a range of structures over the repeats. As an example, the DNA molecule (Figure 2.3a) is also a polymer that repeats the center unit, each connecting to an R-group to define variable nucleotide. The size of the polymer in some molecules is also defined (Figure 2.4). This simplifies the visualization, as in certain scenarios these could be very long molecules.

Given this diversity, there are several exchange formats for the representation of chemical molecules. Some formats are capable to define most of the chemical properties, while others are limited to some properties or can only represent sub-sets of defined molecules. Since the basic structure of a molecule can be a single graph that connects atoms to each other, connection tables are simple exchange formats that describe the molecule by listing all connections between the atoms and their properties. The Mol file is a popular format based on connection tables. It separates atoms and bonds in different blocks, each with

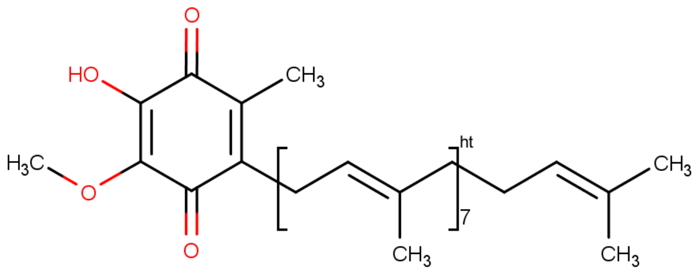


(a) A single stranded DNA molecule.

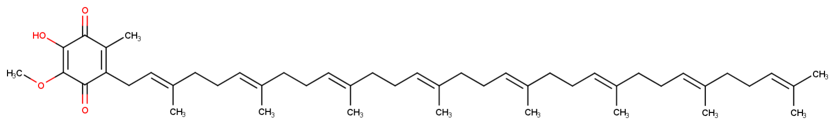


(b) Malonyl-ACP

Figure 2.3: a) A single stranded DNA molecule. Contains one repeating unit at the center, with several R-groups (nucleotides). b) Malonic acid attached to acyl carrier protein.



(a) Defined polymer with 7 repeating units.



(b) The exact representation of the structure.

Figure 2.4: Defined polymer representation and its extended version.

extra columns to define the atom and bond type, but also other chemical properties of the atoms (e.g., charged atoms) and bonds (e.g., sterochemistry). The Mol tables also define atom coordinates, allowing to store the relative position of the atoms for rendering tools to draw molecules. The lack of coordinates would require the use of layout algorithms. For complex molecules most of the layout algorithms fail to properly shape the spatial location of the atoms.

The Chemical Markup Language (CML) is an XML based markup language to exchange compounds. It is more human readable since all attributes are enclosed within tags and it provides many options to store metadata (e.g., titles, comments, text formatting), but also other relevant chemical properties such as spectral data.

A limitation of connection tables is the searching capability, since they are bulky, but also not practical to share or for human interpretation.

The line notation formats are methods to represent compounds in a single string line. This allows for an easier display, limited readability, and for searching purposes.

The InChI [53] string starts with the "InChI=" prefix, followed by the version number. Then it contains six layers separated by the symbol "/" as separator (main / charge / stereochemical / isotopic / fixed-h / reconnected layers).

Because of the detail of the InChI, the size of the string can go up to more than 500 characters, and the string uses a lot of numbers and symbols, in which are unfriendly to search engines. To address this problem the InChIKey was created. The InChIKey is a fixed 27 character string, that is generated from a InChI by using an hashing algorithm function (the SHA-256) of the InChI string [122]. This string is just used for searching and indexing purposes, since there is a collision probability, even though it is quite low.

The InChIKey also allows to rapidly compare two molecules in three distinct levels. The string contains three blocks that are hashed from the connectivity, stereochemistry and protonation layers. The first block is the connectivity layer, a mismatch implies that two molecules have a different skeleton. The second block is the stereochemistry, two InChIKey's with identical first block, but with a distinct second block would imply different stereochemistry. The last block is a single character that matches the protonation: the N character is used for the neutral state, then it moves backwards (M, L, K, etc) or forward (O, P, Q, etc) up until twelve letters to define less or more protons. For a value higher or lower than twelve protons the A character is used.

Since the InChIKey is an hash equality, it does not mean the original InChI is equal, and neither can it be recovered from the InChIKey.

The Simplified Molecular Input Line Entry System (SMILES) [123] representation was created in 1986 at the US Environmental Research Laboratory and later developed and maintained by Daylight Chemical Information Systems, Inc [2]. A community effort was created to give an official definition of the SMILES known as OpenSMILES [94]. The SMILES string allows simple definition of chemical structures by computing a spanning tree over the structure graph, which only stores atoms and bond types. An advantage of the SMILES format is human readability, while the major drawback is the non canonical representation of molecules [92].

Both InChI and SMILES require auxiliary files or rendering algorithms to generate atom coordinates to draw the molecules.

2.2.2 Representing Reactions

The representation of reactions is far more simple compared to the molecules. The common method to define a reaction is just to describe its stoichiometry. While the molecular structure describes the identity of a molecule, the identity of a reaction is defined by two sets of compounds to describe the reactants and the products. The detailed reaction action is described by the bond formation between the reactants and products.

The SMIRKS representation is an exchange format based on SMILES to describe reaction transformations. It allows to encode the transformation between the reactants and the products.

Some aspects may compromise the identity of a reaction. The protonation state of the compounds, which may imply adding or removing proton molecules to balance the

[2]http://www.daylight.com/smiles/index.html

KEGG Reaction: 2.7.1.40 (R00200)

$$\text{ATP} + \text{Pyruvate} \longleftrightarrow \text{ADP} + \text{Phosphoenolpyruvate}$$

MetaCyc Reaction: 2.7.1.40 (PEPDEPHOS-RXN)

$$\text{ATP} + \text{Pyruvate} \longleftarrow \text{ADP} + \text{Phosphoenolpyruvate} + \text{H}^+$$

MetaCyc Reaction: 1.10.3.14 (RXN0-5266)

$$\text{O}_2 + 4\,\text{H}^+ + 2\,\text{an ubiquinol}_{\text{membrane}} \longrightarrow 2\,\textit{a}\ \text{ubiquinone}_{\text{membrane}} + 2\,\text{H}_2\text{O} + 4\,\text{H}^+_{\text{periplasm}}$$

KEGG Reaction: 2.7.7.7 (R00375)

$$\text{dATP} + \text{DNA} \longrightarrow \text{Diphosphate} + \text{DNA}$$

Figure 2.5: The same reaction may have addition of protons due to the protonation state of the participating compounds. Reactions may have the same compound in both sides of the equation due to transport mechanisms. Polymerization reactions may have the same compound in both sides, but representing different polymer units.

stoichiometry, makes the same reaction different in two different conditions.

Lumped reactions are used in some cases where the intermediates of a bioconversion are not known (or not interesting for a given purpose). In these cases, a single reaction may be used to summarize the entire set of conversions into a single reaction. Generic or abstract compounds may describe abstract reactions, that represent combinations of biochemical transformations between the family of the abstract compounds. Unlike metabolites, in a biological perspective, the main interest to catalog reactions is also to associate them with biological enzymes.

The reactions can be either metabolic or transporters (Figure 2.5). A transport reaction moves a metabolite between cellular spaces (compartments) or onto the outside of the cell, while metabolic reactions are other biochemical transformations. This characterization is not that simple since some metabolic reactions may involve transferring a compound between one cellular membrane to another or performing metabolism in two distinct cellular locations (i.e., mitochondria and cytoplasm).

In metabolic databases, most reactions are metabolic, since they represent the reaction action of the enzymes not their actual activity within the cells since these can vary between organisms. Therefore, in many cases the compartment of a transport reaction is an abstract representation (such as *inside/outside* that can be either related to the cell it self or to an organelle, e.g., mitochondria).

Another important aspect are the polymer compounds that were described earlier, since a polymerization reaction may contain the same compound in both sides of the equation. Here, we define basic reactions as those that have non repeated compounds in both sides of the equation. These reactions represent most of the biochemistry in databases. For modeling purposes, in the next chapter, additional reaction classifications are specified.

2.2.3 Biochemical Databases

Currently, the database list from the Nucleic Acid Research journal contains 36 active databases for metabolic pathways, 21 of small molecules and 14 carbohydrates databases[3], while some of the databases share more than one category. These numbers are quite high.

For the purposes of metabolic engineering, the pathway and organisms databases are the ones of higher interest since these organize information around biological entities, connecting biochemistry to organisms. Other databases specialized for certain metabolites are relevant to provide finer detail about the chemical attributes of molecules.

The Kyoto Encyclopedia of Genes and Genomes [63] (KEGG) and MetaCyc are both metabolic and genomic databases since they provide detailed information about metabolic pathways of organisms.

KEGG is sub divided into several databases that are connected to each other, each of these specialized into a specific topic. The KEGG LIGAND collection contains the databases KEGG COMPOUND, GLYCAN, REACTION, RPAIR, RCLASS and ENZYME.

[3]Nucleic Acid Research journal database catalog: `http://www.oxfordjournals.org/nar/database/cap/`

These databases define the general biochemistry of metabolic enzymes. The KEGG COMPOUND contains most of the small molecules that participate in the KEGG REACTION database, a few specialized carbohydrate reactions use the KEGG GLYCAN molecules. There is few redundancy between these databases.

The KEGG MEDICUS contains a set of databases oriented to health and pharmaceutical data. The KEGG DRUG database is a compound database specialized for drug products in Japan, USA and Europe.

MetaCyc [16] is a database from the BioCyc consortium, where unlike KEGG, the databases are specialized for a particular organisms (e.g., EcoCyc - *Escherichia coli*, YeastCyc - *Saccharomyces cerevisiae*, HumanCyc - *Homo sapiens*). MetaCyc is a generic database that contains metabolic information adapted to all biological domains.

The Biochemical, Genetic and Genomic (BiGG) knowledge base provides pathway information based on genome-scale metabolic models. The first version of BiGG (now referred as BiGG1) was published in 2011.

The SEED platform [7] is a model reconstruction tool that allows to easily scaffold genome-scale metabolic models from organisms complete genome, but it requires a library of chemical reactions such as KEGG to assemble the network. This led the creation of the ModelSEED database that is built based on a integration of KEGG, MetaCyc, BiGG (version 1) additionally with a few genome scale models. The main purpose of the ModelSEED database is to provide a dedicated resource for genome-scale metabolic model reconstruction.

The BiGG and ModelSEED databases are both oriented for genome-scale metabolic modeling, since both databases contain information for cellular compartments in the stoichiometry of the reactions, transporter reactions and references to other models.

The Chemical Entities of Biological Interest [47] (ChEBI) contains many thousands of detailed and curated chemical compounds with a chemical ontology, which later was aligned with the Open Biomedical Ontologies (OBO) and Gene Ontology (GO) classifiers. The purpose GO terms is to align gene function to a standard framework allowing cross references between databases.

The LIPID MAPS Structure Database [118] is a database dedicated to lipid structures. Its motivation is to provide proper representation of lipids. Since lipids may have several long carbon chains, it makes difficult for algorithms to draw a proper chemical representation. The database also implements a lipid classification system that splits the compounds into eight groups (fatty acyls, glycerolipids, glycerophospholipids, sphingolipids, sterol lipids, prenol lipids, saccha- rolipids and polyketides). The database contains over 10,000 lipids and with their complete chemical representation.

The Human Metabolome Database [124] (HMDB) and the Yeast Metabolome Database [61] (YMDB) are dedicated databases to the metabolome of the human body and the baker's yeast (*Saccharomyces cerevisiae*), respectively. HMDB contains more than 40,000 metabolites since it catalogs every known compound that can be found in the human body. The main purpose of the database is to provide spectroscopic information about these metabolites for metabolomics studies.

The endless number of biochemical databases provides a rich pool of knowledge to study biological pathways, however the information is scattered between databases and exploring every source is unpractical and time consuming. In this study, the most relevant databases regarding to metabolic modeling are taken into account (Table 2.1). The mechanisms of data access are covered in the implementation section.

2.3 Integration System Design

2.3.1 Introduction

The integration of data has been motivated by the need to provide a centralized unified access to view multiple data sources, such as enterprise information systems, that enables the integration and analysis of business processes, usually with the purpose to support business intelligence for decision making.

The data integration process can be defined by four tasks: data understanding, standardization, specification and execution [45]. To study the data sources is the first task

Table 2.1: Summary of databases related to biological compounds and reactions. M - Contains metabolites. R - Contains reactions. ⋆ - included in this study.

Database	URL	Data	Type
KEGG ⋆	www.genome.jp/kegg	All domains of life	M, R
MetaCyc ⋆	metacyc.org	All domains of life	M, R
ChEBI	www.ebi.ac.uk/chebi	Biological Compounds	M
LipidMAPS ⋆	www.lipidmaps.org	Lipids	M
HMDB ⋆	www.hmdb.ca	Human Metabolome	M
YMDB	www.ymdb.ca	Yeast Metabolome	M
BiGG1 ⋆	bigg1.ucsd.edu	Genome-Scale Models	M, R
BiGG ⋆	bigg.ucsd.edu	Genome-Scale Models	M, R
ModelSEED ⋆	modelseed.org	Integrated	M, R
MetRxn	metrxn.che.psu.edu	Integrated	M, R
MNXRef ⋆	www.metanetx.org	Integrated	M, R
BKM-react	bkm-react.tu-bs.de	Integrated	R

for data integration. The data profiling exercise involves the discovery and extraction of the shape and meaning of the data [91]. Heterogeneous data sources may share identical attributes, but they may also have different meanings.

Also, it is necessary to capture the data types and missing values. The identification of classes is an important step to understand the type of objects of each resource and their hierarchy. Given the previously mentioned resources, some of their attributes may be common to all.

The design of an integrated schema allows to decide how to represent the data sources in an unified view. This requires the identification of the standard representation methods and which attributes to include or transform for the standard representation format.

Data warehouses are databases that usually consolidate distinct heterogeneous data sources into a single unified schema. These tasks are done with the aid of an Extract-Transform-Load (ETL) tool to populate the system. ETL tools usually consist of a set of program scripts that perform tasks to reshape information from several sources into the data warehouse. These tools are not exclusive to tasks related to data cleansing and conformity, but they also include other relevant mechanisms related to data maintenance, such as reporting and scheduling for periodic refresh of the information.

An alternative to this approach is to create virtual data systems. Usually, this is done by implementing a query mediator system, which translates user queries to source queries. This allows to assemble an integrated view without the requirement to materialize data.

Other systems use a middleware to translate messages through the third party data sources, or just to perform routing (if all systems use same protocol).

A common problem to all of the mentioned approaches is the schema and system heterogeneity. These implementation approaches only solve technical and architectural problems regarding performance, maintenance and flexibility of these systems, while data conformity must be unavoidably dealt with domain specific human expertise. Thus, the domain of the information system has a huge impact in the design of the integration methods.

The proposed integration system follows the data warehousing approach, since the

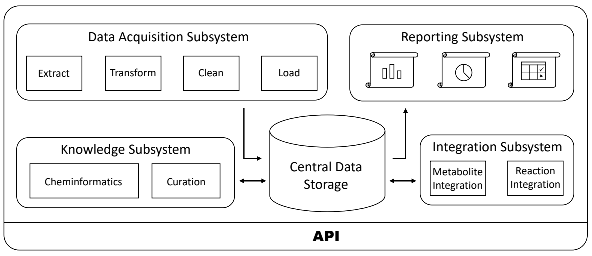


Figure 2.6: Integration system overview. All modules operate with the central storage unit.

purpose of the system is to be capable of generating an integrated catalog of biochemical molecules and reactions. To build an integrated knowledge base of biochemical reactions, an integration system is designed to be able to perform several necessary tasks.

The design of the system contains four subsystems: Data Acquisition, Knowledge, Integration and Reporting (Figure 2.6). The core component is the central data storage (CDS) module. The CDS stores all data collected from the source databases. Data in the CDS are subject to cleansing and conforming processes, but no entity resolution is applied since the main purpose of the CDS is to catalog all records of the source in a common data space.

The data acquisition subsystem is a pipeline composed of several modules that execute the extract, translate, load (ETL) tasks to shape data to the standards of the CDS. It is the entry point of all data in the CDS.

The knowledge subsystem is responsible for the additional manipulation of existing data in the CDS. As a general policy, information stored from the ETL process can not be changed, but noisy or bad records can be flagged. This allows to track issues from source databases, but also to execute cleaning and fixing tasks.

The reporting subsystem generates several types of reports, which allow to track the content of the CDS. Finally, the integration subsystem provides the entity resolution methods to identify duplicates in the CDS. Their subsystems are covered in the following sections.

2.3.2 A Generic Metabolic Repository

In this section several rules are defined to build the CDS module to be flexible but consistent. The database should be able to accommodate new domains in the future, and be able to host different shapes of instances, not limited to metabolites or reactions, but to be expanded, for instance to genes, genomes and proteins.

Before describing the database rules, a general definition of the symbols that will be used is provided:

Calligraphy typing	$\mathcal{M}, \mathcal{R}, \mathcal{E}$	Special Sets
Capital letters	M, R, S	Sets
Lowercase letters	m, r, e, o	Variables or Objects

The square brackets $\langle a, b, c, d\rangle$ are used to define ordered tuples, while parenthesis (a, b, c, d) represent the unordered tuples. The π_n function returns the *n-th* element of an ordered tuple. A dictionary (with function) is defined by $\{\}$. As an example, given a dictionary $dict = \{attr : value\}$, then the key value pair is addressed as $dict[attr] = value$. The $\{\}$ is also used as the set builder notation (e.g., $\{x \mid x > 0\}$).

For simplification purposes the special sets are also used as functions, as an example given a graph defined by $\mathcal{G} = \langle \mathcal{V}, \mathcal{E}\rangle$, then $\mathcal{V}(\mathcal{G})$ is read as the $\mathcal{V}$ set of the $\mathcal{G}$ tuple (in this case, $\mathcal{V}(\mathcal{G}) = \pi_0(\mathcal{G})$).

The top level of the database is defined by the set of namespaces. Every object in the system must belong to a unique namespace, which is defined within the $\mathcal{N}$ domain (Example 1), where $\mathcal{N}$ is the set of all namespaces.

Example 1. *Namespaces*

Consider a system that defines objects of two distinct databases ($Database_1$, $Database_2$) having two types of properties (Formula, InChI). Then $\mathcal{N}$ = ($Database_1$, $Database_2$, Formula, InChI) would be the namespace domain of the system. No other properties or database namespaces may be present in this system.

The namespace is an abstract label that groups sets of objects in the database. An object (Definition 1) in the database can represent any entity of interest. Each object must belong to a single namespace, although objects can be replicated into different namespaces, for instance to spawn different versions of the same database.

Definition 1. *Object*

An object is an ordered tuple $o = \langle id, n, a \rangle$, such that $n \in \mathcal{N}$, id is a string identifier, a is a set of arbitrary attributes of o.

The object represents an entity, that can be either a metabolite or a reaction, but also properties such as a formula or an InChI (Example 1). The meaning of the object is given by the namespace that it belongs to. The set $\mathcal{V}$ is the set of all objects in the system.

A relationship (Definition 2) is an edge between two distinct objects.

Definition 2. *Relationship*

A relationship is a directed edge represented by an ordered tuple $e = \langle o_1, o_2, t, a \rangle$, such that $o_1, o_2 \in \mathcal{V} \times \mathcal{V}, o_1 \neq o_2$, t is the type of the relationship, and a is a set of arbitrary attributes of e.

The interaction e between two objects is defined by $t(e) = \pi_3(e)$, where t is the type of the interaction, such as an ownership (e.g, *has_formula*, *has_inchi*, etc), a relationship (e.g, *has_crossreference_to*, *instance_of*, etc), etc. As an example, the ownership of o_b to o_a is defined as $e_{a,b} = \langle o_a, o_b, has\_an, \{\} \rangle$. For simplification purposes, t and a are omitted when not relevant ($e_{a,b} = \langle o_a, o_b \rangle$).

Together, all objects and edges assemble an universal graph $\mathcal{G}$ (Definition 3), which is the universal database for metabolites and reactions.

Definition 3. *Universal Graph*

Let $\mathcal{V}$ be the set of all objects, $\mathcal{E}$ the set of all edges, the universal graph $\mathcal{G}$ is defined as $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$

Duplicate identifiers can coexist in different namespaces (Definition 4), while objects within the same namespace are not allowed to share the same identifier. Therefore, the unique identifier is the pair $\langle id, namespace \rangle$ composed by an identifier and a namespace (Example 2).

Definition 4. *Uniqueness*

Let $o_a, o_b \in \mathcal{V}$*, such that,* $o_a = \langle id_a, n_a, attr_a \rangle$ *and* $o_b = \langle id_b, n_b, attr_b \rangle$*, then* $o_a = o_b$ *if and only if,* $id_a = id_b \wedge n_a = n_b$*.*

Example 2. *Unique Objects*

Given two objects $o_1 = \langle 12345, PubChem, \{\} \rangle$ *and* $o_2 = \langle 12345, ChEBI, \{\} \rangle$ *although they have the same identifier* 12345*, they are distinct objects since* $PubChem \neq ChEBI$*.*

Instances in the database can be defined as a proxy. The function $proxy : o \rightarrow Boolean$ defines if an object is a proxy. Proxies are *maybe* objects that mark temporary placeholders for future instances, since they may exist or not. They are essential to deal with database referencing, since many databases reference others, but the actual information about the references is only revealed if they are loaded afterwards from the original source. In some cases, it is also possible that, for certain references, the referenced records may not exist due to update cycles or referencing errors.

The universal graph $\mathcal{G}$ is actually a set of subgraphs, such that each defines a specific domain of instances (i.e., metabolites, reactions). The metabolite graph $\mathcal{G}_m$ (Definition 5) is the subgraph that defines the entire metabolite domain.

Definition 5. *Metabolite Graph*

Let $\mathcal{M} \subseteq \mathcal{V}, \mathcal{P}_m \subseteq \mathcal{V}$ *and* $\mathcal{E}_m \subseteq \mathcal{E}$*, the metabolite graph is defined as* $\mathcal{G}_m = \langle \mathcal{M} \cup \mathcal{P}_m, \mathcal{E}_m \rangle$*.*

The subgraph $\mathcal{G}_m$ defines all objects that represent metabolite instances $\mathcal{M}$ and their property instances $\mathcal{P}_m$ together with their relationships $\mathcal{E}_m$.

The graph $\mathcal{G}_m$ is not bipartite since cross-referencing is represented by edges that connect two metabolites. Properties $p \in \mathcal{P}_m$ also are allowed to be connected to each other to define relevant relationships between biochemical properties (e.g, equivalent structure formats).

A reaction entity works the same as the metabolite as they also must belong to a unique namespace. The stoichiometry of the reaction is defined by having an edge between the reaction object and the metabolite object (Example 3) and the value is assigned in the attributes of the edge.

Example 3. *Stoichiometry*

Let r be a reaction with the following stoichiometry $a + b \longrightarrow 2\ c$.

The stoichiometry would be defined with 3 edges $e_a = \langle r, a, left, \{value : 1\}\rangle$, $e_b = \langle r, b, left, \{value : 1\}\rangle$, $e_c = \langle r, c, right, \{value : 2\}\rangle$

The reaction graph $\mathcal{G}_r$ (Definition 6) is the subgraph that defines the reaction's domain.

Definition 6. *Reaction Graph*

Let $\mathcal{M}, \mathcal{R}, \mathcal{P}_r \subseteq \mathcal{V}$ and $\mathcal{E}_r \subseteq \mathcal{E}$, the reaction graph is defined as $\mathcal{G}_r = \langle \mathcal{M} \cup \mathcal{R} \cup \mathcal{P}_r, \mathcal{E}_r\rangle$.

Like in $\mathcal{G}_m$, $\mathcal{R}$ is the set of all reaction objects and their properties $\mathcal{P}_r$. The relationship between $\mathcal{M}, \mathcal{R}, \mathcal{P}_r$ is defined in $\mathcal{E}_r$.

The subgraphs $\mathcal{G}_m$ and $\mathcal{G}_r$ together make the universal graph $\mathcal{G}$, where $\mathcal{V} = \mathcal{M} \cup \mathcal{R} \cup \mathcal{P}_m \cup \mathcal{P}_r$ and $\mathcal{E} = \mathcal{E}_m \cup \mathcal{E}_r$.

The universal graph $\mathcal{G}$ can be easily expanded to other domains of objects. In the next chapter, the metabolic models will be introduced in the system, but also for future reference, additional logical domains can be added such as genes and taxonomy.

2.3.3 Data Acquisition

The data acquisition subsystem is perhaps the bottleneck for expanding the integration database, since each external resource requires dedicated methods to extract relevant information. The study and understanding of the domain logic of external databases is also a manual and time consuming process because of schema heterogeneity.

In addition to these issues, another problem that integration systems have to tackle is the syntactical heterogeneity of the third-party resources. Terms may have different names in each of the external resource and it is important to capture which distinct terms are synonyms and which equal terms are actually non-equal.

The ETL pipeline requires three components, that two are specific for each individual data source to be included in the CDS. The extract component is responsible to list and pull instances from the data source. This depends on the physical location of the data, be either local files or web API's.

The transform component is responsible for transforming the extracted object into a graph. At this stage, the object must conform to the standards of the CDS, which implies the addition of the namespace tag and dissecting the properties into graph vertices.

Finally, the load component merges the extracted graph with the universal graph database. This step is generic since all objects are already reshaped to fit the CDS rules.

Extract. Third-party data systems have their own specific methods and rules to access the database information. The extraction phase is responsible for querying these systems.

External sources can be any stream of information, such as local files, web interfaces, database systems, etc. For each external resource, it is necessary to develop or integrate an extraction component.

The extraction component is responsible for two important functions: *list* and *fetch* the records. The list function is responsible for returning a set of unique identifiers, which must be valid for the fetch operation to pull from the database. The fetch operation produces a raw entity e, that at this stage does not need to obey any rules of the integrated data storage.

Transform. The transform function $\mathcal{T} : e \to \mathcal{G}'$ maps native e instances into graph objects $\mathcal{G}'$. Entities from the extract phase are transformed into a graph $\mathcal{G}' = \langle \mathcal{V}', \mathcal{E}' \rangle$. The graph $\mathcal{G}'$ is usually star shaped, containing entities and properties objects with their respective relationship edges.

Usually a single $o \in \mathcal{V}$ is non proxy (the entity itself) while the remaining are proxy entities corresponding to the references of the extracted entity e.

The transform function is responsible for giving the namespace to each $o \in \mathcal{V}'$. This would allow later to detect if these objects are already present in the CDS.

Cleanse. Minor corrections are performed to conform a few of the properties. This includes the translation of a few names, such as databases names (deduced from the initial profiling step) into a controlled vocabulary that is in conformity with the global schema. Chemical formulas are also corrected to a standard representation to avoid duplicates. It is common that in a few cases the system is unable to translate or read properties because of bad or invalid representation. In these scenarios, a warning is issued and logged, while no correction is performed. Instances of each resource are reshaped to graph structures. The properties of interest are translated into edge semantics.

However, the presence of resource specific attributes is also found in several of these databases. The question is whether these attributes plays a significant role in the integration of the instances. As an example, the KEGG Ligand Compound database is referred using distinct names in other resources, such as *LIGAND-CPD* or *KEGG COMPOUND accession* in MetaCyc and ChEBI respectively. This problem also applies to attributes and other properties. The understanding of third party domains is crucial to avoid future problems that may compromise the integrity of the system, although later it is still possible to fix and fuse the redundant classes. Another purpose of this task is to decide whether to include the data source or to discard it in the system.

Load. Loading instances to the CDS is done by merging the $\mathcal{G}' = \langle \mathcal{V}', \mathcal{E}' \rangle$ from $\mathcal{T}$ with the main graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$. New objects are transfered to the graph, while for objects present in both $\mathcal{V}'$ and $\mathcal{V}$, the attributes of $o \in \mathcal{V}'$ are transferred to the previous instance in $\mathcal{V}$ only if o is not a proxy. Then, the new edges in $\mathcal{E}'$ are all transferred to $\mathcal{E}$.

2.3.4 Knowledge expansion

The cheminformatics module belongs to the knowledge subsystem, being an optional step after the ETL pipeline and before the integration methods. The purpose of this module is to apply computational methods to analyze and generate additional chemical data, such as equivalent structures to exploit the interchangeability of chemical exchange formats [93].

By generating alternative chemical representations from the existing metabolite properties, it allows to fill missing equivalent representations, but also to add relationships between metabolite properties (e.g., a formula that is related to an InChI). It also allows to perform structural validation between the structures annotated in the molecules, since given a set of properties of a molecule, it is expected that they are related to each other.

The interchangeability of exchange formats is not always bidirectional. For instance, an InChI is usually convertible to a SMILES representation, but not all SMILES may be converted into an InChI. As an example, the generic R-groups are not supported by the InChI representation, but are allowed in SMILES.

Due to the non canonical representation of the SMILES, an additional standardization of SMILES structures is advised. The universal SMILES method [92] takes advantage of the InChI canonization strategy to generate canonical SMILES. If a SMILES transforms to itself, from the canonization method, it is considered universal.

The names of chemical compounds are usually referred in the literature as weak descriptors to identify molecules, due to the ambiguity caused by the several synonyms. Structure to name conversion is possible using the IUPAC naming system that allows to name organic molecules following the same philosophy of the InChI structure. The purpose of this method is to implement a canonical name generating strategy, following the InChI principles. The name to structure methods allow to convert chemical names into structures by using a controlled vocabulary, allowing to connect or generate their hypothetical structure.

Other properties, such as formula and InChIKey are also generated from the molecular structure. While they are unable to give the structural identity of a molecule, these properties provide comparison methods to assess the similarity confidence.

The InChIKey hashes the InChI layers into three distinct blocks: the first includes atomic content and the connectivity layer, while the second hash block is related to the stereochemistry layer, and the last block is a single character, where **N** (neutral) stands for the neutrally charged molecule and by increasing and decreasing the charge, the character changes. Molecules with equal connectivity will share the first hash block, allowing for easy comparison and grouping of compounds with similar connectivity or protonation states

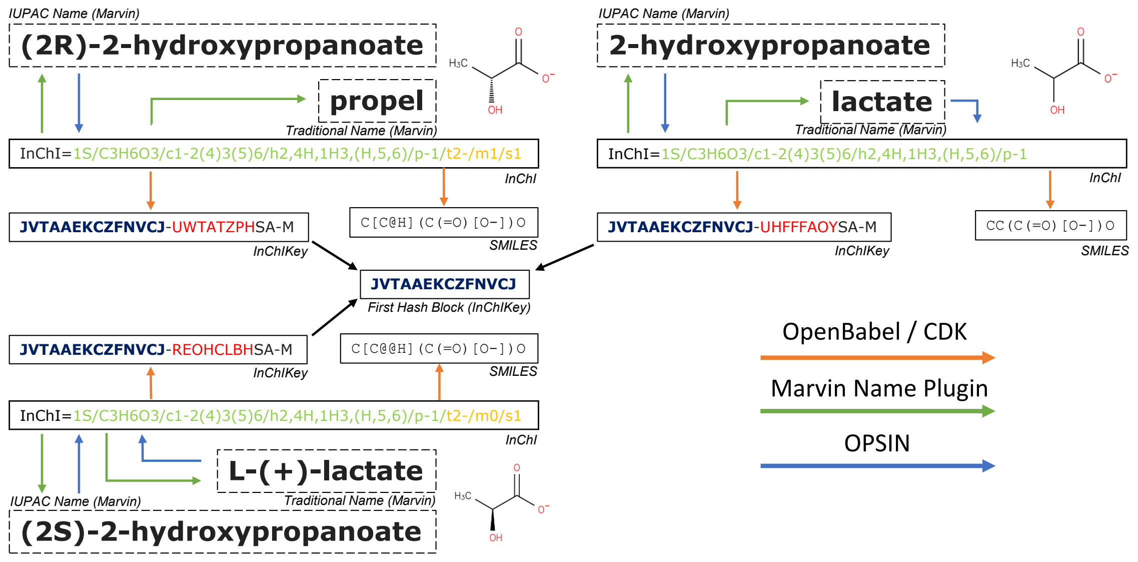

IUPAC Name (Marvin)
(2R)-2-hydroxypropanoate
propel
Traditional Name (Marvin)
InChI=1S/C3H6O3/c1-2(4)3(5)6/h2,4H,1H3,(H,5,6)/p-1/t2-/m1/s1
InChI
JVTAAEKCZFNVCJ-UWTATZPHSA-M
InChIKey
C[C@H](C(=O)[O-])O
SMILES
JVTAAEKCZFNVCJ
First Hash Block (InChIKey)
JVTAAEKCZFNVCJ-REOHCLBHSA-M
InChIKey
C[C@@H](C(=O)[O-])O
SMILES
InChI=1S/C3H6O3/c1-2(4)3(5)6/h2,4H,1H3,(H,5,6)/p-1/t2-/m0/s1
InChI
L-(+)-lactate
IUPAC Name (Marvin)
Traditional Name (Marvin)
(2S)-2-hydroxypropanoate
IUPAC Name (Marvin)
2-hydroxypropanoate
lactate
Traditional Name (Marvin)
InChI=1S/C3H6O3/c1-2(4)3(5)6/h2,4H,1H3,(H,5,6)/p-1
InChI
JVTAAEKCZFNVCJ-UHFFFAOYSA-M
InChIKey
CC(C(=O)[O-])O
SMILES
OpenBabel / CDK
Marvin Name Plugin
OPSIN

Figure 2.7: Lactate stereoisomers chemical properties and their relationships. Properties are connected to each other based on the output of the cheminformatics tools.

(Figure 2.7).

For integration purposes, the main properties are the universal SMILES, InChI for identity purposes, while InChIKey and chemical formulas are valuable for match confidence.

The expanding process is iterated and stops when every possible property is generated and connected to related properties. The terminal properties are all of the properties (Table 2.2) that are only generated (i.e., chemical formula, InChIKey), being the order to scaffold additional properties as follows: Mol Format; Name; SMILES; InChI.

From a structural point of view, the chemical structure dictates the identity of a molecule. A clustering of the properties allows to detect inconsistencies in the database instances. In general, a single cluster of properties should be related to each molecule. However, multiple clusters are also allowed if they only differentiate in the protonation of the structure.

Table 2.2: Property conversion table. I - InChI, K - InChI Key, S - SMILES, U - Universal SMILES, n - Name, N - IUPAC Name, F - Molecular Formula, M - Mol Format

From \ To	I	K	S	U	n	N	F	M
I		X		X			X	
K								
S	X			X			X	
U	X						X	
n						X		
N	X			X			X	
F								
M	X			X			X	

2.4 Integration

2.4.1 Biochemistry Integration

The integration pipeline is a cyclic process that may take many iterations. Staring with metabolite clustering, this first step of the pipeline merges equivalent metabolite instances of the $\mathcal{M}$ domain.

The second step unifies reaction instances, where the reconciliation of reactions is solely dependent on the metabolite integration since their identity is based on the stoichiometry.

The following steps extract useful knowledge from the reaction sets, a common practice in other reconciliations is to extract metabolite similarity from the unpaired metabolites of the partial matching formulations [74, 9].

Finally, before the next iteration user input and evaluation is used to refine the next cycle, and declustering methods may also be applied for a more conservative integration.

The product of the integration pipeline are reference sets that merge database references similar to the MetaNetX unified reference space [90]. The assembly method materializes the references into a consensus record that unifies the properties and attributes of the clusters.

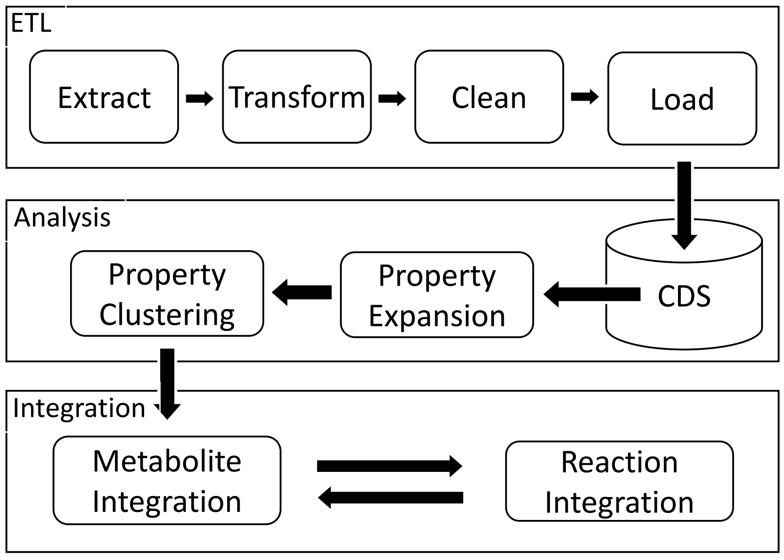


Figure 2.8: Integration pipeline. The ETL pipeline populates the CDS, followed by the knowledge expansion module to analyze and extend the chemical properties. In integration of metabolites and reactions takes place at the end as an iterative cycles between both.

2.4.2 Integration of Metabolites

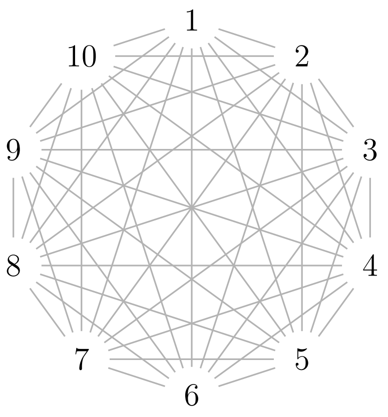
Integration of metabolites takes place by clustering all metabolite instances of $m \in \mathcal{M}$ into sub-sets $M_i \subseteq \mathcal{M}$, where each is an integrated metabolite set, i.e. represents information on a simple metabolite, or ideally seeks to do so.

The clustering rules of the instances are defined by the purpose of the integration. In the following methods, two given metabolites are considered duplicates if they represent the same molecule. This implies that the different protonation states of compounds are considered to be equal.

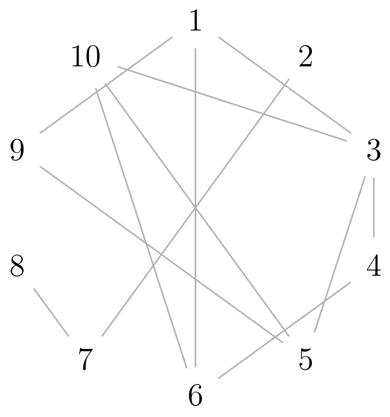
The integration is calculated by pairwise entity resolution, i.e. merging pairs of entities, between all possible metabolites and assigning those numerical values (weights). A similarity function $\phi : (a, b) \mapsto \mathbb{R}$ is defined, where $a, b \in \mathcal{M}$, that computes the similarity score between two metabolites.

Definition 7. *Identity* $\equiv$

Let m_1, m_2 be two metabolites in $\mathcal{M}$ and $\omega \in \mathbb{R}^+$ a constant value, if $\phi(m_1, m_2) > \omega$,



(a) Γ: all possible duplicates.



(b) Γ_ϕ: connected metabolites are considered to be equivalent

Figure 2.9: Metabolite clustering by pairwise matching.

then it is said that m_1 is identical to m_2, such that $m_1 \equiv m_2$.

The ω is a defined threshold to be set by the integration engine, defining the minimum similarity score to consider two metabolites as duplicates. According to identity rule (Definition 7), the challenge is to define the true ϕ function and optimal value for ω.

The set Γ (Equation 2.1) defines the space of all pairs of metabolites in $\mathcal{M}$. Γ_ϕ is a subset of Γ, being the space of all pairs of metabolites that are duplicates filtered by ϕ and ω.

$$\Gamma = \{(a, b) \in \mathcal{M} \times \mathcal{M},\ a \neq b\} \tag{2.1}$$

$$\Gamma_\phi = \{(a, b) \in \Gamma \mid \phi(a, b) > \omega\} \tag{2.2}$$

The Γ set is a complete graph (Figure 2.9a) with all metabolites in $\mathcal{M}$. Independently of how many data sources are in $\mathcal{M}$, every instance is subject to deduplication.

Each metabolic database may catalog thousands of instances and, thus, in a complete graph the number of edges increases exponentially $\frac{n(n-1)}{2}$ for n vertices. It is computationally infeasible to calculate the score for all pairs given the size of $\mathcal{M}$.

For most of the edges e in Γ, it is likely that $\phi(e) \leq 0$ since most metabolites are totally unrelated to each other (Figure 2.9b).

The **FilterGammaSubset** algorithm (Algorithm 1) computes the subset Γ' of Γ that contains only pairs of metabolites that might have at least one relationship in common. The formula property is an exception and it does not count as an interaction, since it cannot by itself define $\equiv$ between compounds. Later, the formula is used to amplify or diminish the overall score of the $\equiv$.

Algorithm 1 Compute Γ subset

```
1: procedure FilterGammaSubset($\mathcal{M}, \mathcal{P}_m, \mathcal{E}$)
   Input: $\mathcal{M}$, $\mathcal{P}_m$, $\mathcal{E}$ (the domain of metabolites, metabolite properties and edges)
   Output: $\langle \mathcal{V}_\Gamma, \mathcal{E}_\Gamma \rangle$ (A graph $\Gamma' \subseteq \Gamma$ )
2:     $\mathcal{V}_\Gamma \leftarrow \emptyset$                                   ▷ Initialize empty vertice set
3:     $\mathcal{E}_\Gamma \leftarrow \emptyset$                                   ▷ Initialize empty edge set
4:     for $e \in \mathcal{E}$ such that $\pi_2(e) \in \mathcal{P}_m$ do
5:         $p \leftarrow \pi_2(e)$
6:         $M \leftarrow \emptyset$                                ▷ Initialize empty metabolite set
7:         for $e \in \mathcal{E}$ such that $\pi_1(e) \in \mathcal{M}$ do
8:             $M \leftarrow M \cup \pi_1(e)$
9:         $E \leftarrow \{\langle a, b \rangle \mid \langle a, b \rangle \in \wp(M)\}$         ▷ All combinations of $a$, $b$ in $M$
10:        $\mathcal{V}_\Gamma \leftarrow \mathcal{V}_\Gamma \cup M$
11:        $\mathcal{E}_\Gamma \leftarrow \mathcal{E}_\Gamma \cup E$
12:    return $\langle \mathcal{V}_\Gamma, \mathcal{E}_\Gamma \rangle$
```

To define the ϕ function, previous integration methods are reviewed to assess the existing strategies for metabolite integration.

Molecular structural analysis should be the most reliable property for duplication detection, which is used by the previous reconciliations (BKM-react, MetRxn, MetaNetX). However, due to the integration goals, molecules that are presented in distinct protonation states should be considered identical, which implies manipulation of the structure format to detect such occurrences. The BKM-react drops the layers related to the ionization state, while the other two databases attempt to transform the molecular structures at a certain pH value, allowing to standardize the InChI's and SMILES's.

The metabolite nomenclatures also contribute to the integration. However, the exact mechanism in most of the integrations is not clearly detailed. The BKM-react states that names and synonyms were combined, while the MNXRef applies an automated reconciliation when matches are found between names and molecular formula. MetRxn uses names for curation purposes, being the names tokenized by highlighting stereo information (e.g., L-/D-, *cis/trans*, etc), or equivalences (e.g., "-ic acid" and "-ate").

The reaction context strategy allows to infer equivalent metabolites by matching the stoichiometry of the reactions. Given two reactions r_1 and r_2, if there is only a single metabolite in the stoichiometry that differs if may hint a potential duplicate. This method was used by MNXRef and MetRxn.

According to Bernard *et al.*[9], none of the previous integrations reported the use of cross-references to integrated the databases (with the exception of the cross-references between reactions in MetaNetX).

The standard similarity function (Equation 2.3) is a definition of ϕ to mimic previous integration methods. The function involves three components: names, structures and reactions. Each of the components are contribution functions that add up to the similarity score, representing different properties used to create and weight links between pairs of metabolites.

$$\phi_{std} = N_{std} + S_{std} + R_{std} \tag{2.3}$$

To mimic the exact approach for each method would be impossible for several reasons:

- It is not known exactly how the names of the compounds contribute for the integration in many of the previous methods.
- It is also not clear which were the exact methods used to compute the InChI sub layers or how the comparison is made.

- The method used to standardize the molecules to a certain pH may involve different tools, such as the commercial software Marvin.
- Most of the original datasets of these integrations are outdated versions which are inaccessible.

To compare to existing methods all of the subcomponents of ϕ_{std} are defined as binary functions (e.g., $S_{std} : \mathcal{M} \times \mathcal{M} \mapsto \{0, 1\}$) and ω is set to be 1, such that m_a and m_b are equal if one of the components is true.

The structure function S_{std} is 1 for every two metabolites that share the same connectivity and stereo structure.

The reaction function R_{std} is the feedback from the reaction integration. The contribution of this function is possibly 1 in the second iteration (and following) of the integration.

The name function N_{std} is set to be undefined, since it is not known how exactly they are defined in the other databases.

The major drawback of the ϕ_{std} is that it lacks the interaction between each component. This implies that each of the contribution functions are independent of each other, and a single function is enough to declare if $m_a \equiv m_b$.

To improve the scoring method, the interaction between the functions is extended to capture negative rules. This allows the scoring function to reject incorrect properties based on the overall contribution of all other properties in the metabolites.

The improved functions are defined as follows:

The function $N : (a, b) \mapsto \mathbb{R}$ (Equation 2.4) defines a name scoring method. The positive name rules count the occurrence of name matches between a and b split into three categories: matches between IUPAC names n_{iupac}, matches between exact strings n_{exact} and matches between similar strings n_{mod}, each with their own weight, $\alpha_n, \beta_n, \gamma_n \in \mathcal{R}$. The IUPAC names should have a higher score of all of the three, while the other should have smaller scores. The matches between modified names only occur because of the transformation and/or cleansing performed in the previous ETL phase. In this work only case insensitive matches are considered. It is known that for certain names (e.g., molecular formulas), the context may change if the string is treated

The negative rule N_{neg} (Equation 2.5) takes into account the number of unmatched names, but only if the number of synonyms n_{count} surpasses a certain $\rho_n \in \mathbb{Z}$ limit. Compounds with only a single name have low match probability, even if they are true equals due to many synonyms. Still, if the maximum number of synonyms increases, the expected odds to have a positive match should be much higher if two compounds are truly equal.

$$N = n_{exact} . \alpha_n + n_{mod} . \beta_n + n_{iupac} . \gamma_n + N_{neg} \tag{2.4}$$

$$N_{neg} = \begin{cases} -\kappa_n & \text{if } n_{count} > \rho_n \wedge (n_{exact} + n_{mod} + n_{iupac}) < 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

To summarize, the N function take five parameters: α_n, β_n and γ_n are similarity weights, the ρ_n defines the number of synonyms to utilize the negative rule, and the κ_n is the penalty for failing the rule.

To evaluate structural similarity both InChI and SMILES are considered, each with a positive (Equation 2.6) and a negative (Equation 2.7) component.

In fact, every molecular exchange format can be considered as a general structural data comparison function $\sum_{stype} S^{+}_{stype} + S^{-}_{stype}$, but since only InChI and SMILES are regularly used, the S functions were defined to cover only these two types ($S^{\pm}_{i}$ for InChI and $S^{\pm}_{s}$ for SMILES).

$$S^{+}_{i/s} = \begin{cases} \alpha^{+}_{i/s} & \text{if an exact match occurs} \\ \beta^{+}_{i/s} & \text{if there is a protonation mismatch} \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

$$S^-_{i/s} = \begin{cases} -\alpha^-_{i/s} & \text{if there is connectivity mismatch} \\ -\beta^-_{i/s} & \text{if stereo mismatch occurs} \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

The positive part scores exact matches and partial matches (distinct protonation levels). The negative parts penalizes different connectivities or stereos mismatches each with individual scores.

$$S = S^+_i + S^-_i + S^+_s + S^-_s \tag{2.8}$$

The final function S (Equation 2.8) sums the contribution of the molecular exchange formats, the $\alpha^{\pm}$ and $\beta^{\pm}$ weights allow to tune the strength of chemical representation in the similarity score.

Two scenarios are considered for cross referencing (Equation 2.9) between databases: if the reference is unidirectional, such that either only M_a references M_b or vice versa, then a lower score value maybe considered, while if both reference each other a higher score can be given.

$$X = \begin{cases} \alpha_x & \text{if reference is bidirectional} \\ \beta_x & \text{if reference is unidirectional} \end{cases} \tag{2.9}$$

The molecular formula $F : (a, b) \mapsto \mathcal{R}$ functions analyse the atomic composition of the two compounds. Fully determined molecular formulas can match exactly (f_{exact}) or only mismatch hydrogen (f_H). Molecular formulas matching with the R-group are given a different weight (γ_f) since it is not trivial to decide whether both R-groups have the same meaning or not. As a special category, some formula strings may fail software parsing (contain invalid atoms or have string descriptions in the middle), but in some cases they are inherited from other databases (these formulas are represented as f_{bad} and have a separate score assigned).

Overall, F is defined as:

$$F(a,b) = f_{exact}.\alpha_f + f_H.\beta_f + f_R.\gamma_f + f_{bad}.\eta_f \tag{2.10}$$

where f_{exact}, f_H, f_R and f_{bad} are the occurrences of exact matches, matches with exception of hydrogen atoms, matches containing R-groups and matches between unparseable formula strings, with their respective weights $\alpha_f, \beta_f, \gamma_f$ and η_f.

The reaction function takes into consideration the reactions where the two compounds a, b participate. In this case, the negative feedback is given if two distinct compounds a and b participate in the same reaction r.

Let r be any reaction in $\mathcal{R}$, let e_1, e_2 be any edges in $\mathcal{E}$, then if

$$\exists_{r\in\mathcal{R}}\exists_{e_2,e_2\in\mathcal{E}}[\pi_1(e_1) = \pi_1(e_2) = r \wedge \pi_2(e_1) = a \wedge \pi_2(e_2) = b \wedge e_1 \neq e_2] \tag{2.11}$$

This constraint implies that either the stoichiometry of r contains the metabolite twice or simply $a \equiv b$ is not true. The first is unlikely to happen, therefore this assertion is used as negative feedback (Definition 2.12) scored by α_r. In fact, this could also be considered a strong negative feedback since merging elements within the stoichiometry of any reaction may imply severe inconsistencies.

The reaction function also receives feedback from the integration of reactions. This is, however, only possible after the first iteration of the integration process since an initial integrated set of compounds is necessary to integrate the reactions. The R function is revised in the next section to define the β_r rule.

$$R(a,b) = \begin{cases} -\alpha_r & \exists_{e_1}\exists_{e_2}[e_1 \neq e_2 \wedge \pi_1(e_1) = \pi_1(e_2) = r \wedge \pi_2(e_1) = a \wedge \pi_2(e_1) = b] \\ \beta_r & \text{if there is reaction integration feedback} \\ 0 & \text{otherwise} \end{cases} \tag{2.12}$$

For curation purposes, users can manually group compounds into sets. This set of compound sets is referred as a curated set. The function C (Equation 2.13) scores instances based on manually assigned groups in the curated set U. The positive feedback is given by the score α_c, when there is a set $M \in U$ that contains both metabolites. A negative score β_c is given if the metabolites are found in two distinct sets in U.

$$C(a,b,U) = \begin{cases} \alpha_c & \text{if } \exists_{M \in U}[a \in M \wedge b \in M] \\ -\beta_c & \text{if } \exists_{M_1,M_2 \in U}[a \in M_1 \wedge b \in M_2 \wedge M_1 \neq M_2] \\ 0 & \text{otherwise} \end{cases} \tag{2.13}$$

The last function $Z(a,b)$ handles the exceptional attributes that may be relevant to identify compound identity (e.g, database specific attributes). For instance, BiGG1 and BiGG are related to each other, therefore they are merged by simple a identifier (i.e., BiGG abbreviation) comparison. A few string transformations also take place since in the recent BiGG database, where the dash character was replaced by double a underscore.

$$\phi = (N + S + X + R + Z + C).(1 + F) \tag{2.14}$$

The integrated metabolite space is defined in the set $\mathcal{I}$ (Definition 8), the final product of the integration method that contains sets of metabolites M that are the integrated metabolite clusters.

Definition 8. *Integrated Metabolites*

$$\mathcal{I} = \{M \in \wp(\mathcal{M}) \backslash \emptyset\}$$

The integrated metabolite domain is actually given by the connected components (CC) of Γ_ϕ. Therefore, the function $CC(\Gamma_\phi) = \mathcal{I}$ performs the clustering of the metabolites.

2.4.3 Integration of Reactions

The reaction properties $\mathcal{P}_r$ are not suitable for integration. Indeed, unlike metabolites, most of the reactions have little to zero attributes. The name and the Enzyme Commission Number (EC number) are usually the only properties found for reactions.

The EC number is a system to classify enzymes based on their biochemical activity. It consists of four positive integers (i.k.j.l), displayed hierarchically. The first number represents the major function of the enzyme, of a total of six classes (Oxidoreductases, Transferases, Hydrolases, Lyases, Isomerases, Ligases). The second level indicates the functional groups that the enzyme is acting upon (e.g, CH-OH groups), while the two remaining numbers characterizes the cofactors (e.g., NAD/NADP) and substrates.

A problem of this classification method is the lack of specificity. As an example, the EC 1.1.1.1 which represents the role of an alcohol dehydrogenase is assigned to more than 10 reactions.

Two reactions are said to be equal if they have equivalent reaction stoichiometry. This implies that reactants and products are the same and the coefficients are also equivalent.

The reversibility of the reaction is defined by several factors. In most studies, the Gibbs free energy of the reaction is used to decide if a reaction is reversible or irreversible. However, enzyme kinetics and substrates concentration may also play a role to decide whether a reaction is bidirectional or unidirectional [4].

Because of this, many databases do not have a defined reversibility for reactions. Therefore, the distinction of product and reactant by assuming the original direction of the reaction is many times ambiguous and pointless. Instead, the reactants and products are defined as left (LHS) or right (RHS) components of the stoichiometry (Definition 9). The assignment of the molecules to either left of right is purely based on the original position of the compounds found in the stoichiometry of the external databases.

Definition 9. *Reaction Stoichiometry*

Let $StoichEdges(r, \mathcal{E}) = \{e \mid (\exists_{e \in \mathcal{E}_r})[\pi_1(e) = r \wedge ns(e) \in \{left, right\}]\}$ *be the set of all stoichiometry edges of a reaction r.*

The stoichiometry of a reaction r is an ordered tuple, where $Stoich(r, \mathcal{E}) = \langle LHS(r, \mathcal{E}), RHS(r, \mathcal{E})\rangle$*, such that:*

$LHS(r, \mathcal{E}) = \{\langle \pi_2(e), value(e)\rangle \in \mathcal{M} \times \mathbb{R} \mid e \in StoichEdges(r, \mathcal{E}) \wedge t(e) = left\}$

$RHS(r, \mathcal{E}) = \{\langle \pi_2(e), value(e)\rangle \in \mathcal{M} \times \mathbb{R} \mid e \in StoichEdges(r, \mathcal{E}) \wedge t(e) = right\}$

where, t(e) is the type of the edge. Both LHS and RHS are ordered tuples of a metabolite and a positive real value.

The stoichiometry is defined by an ordered tuple containing two unordered sets. The $Stoich(r, \mathcal{E})$ function (Definition 9) assembles the tuple from the edges $\mathcal{E}$. For simplification, in the $Stoich(r, \mathcal{E})$ function, the global parameter $\mathcal{E}$ is omitted, and the function call is rewritten as $Stoich(r)$.

The equality of two reactions if given by the equality function ($\simeq$), which is true if there is a match between both left and right sets of their stoichiometry.

Definition 10. *Reaction Equality*

Let $s(r_a) = \langle LHS_a, RHS_a \rangle$ *and* $s(r_b) = \langle LHS_b, RHS_b \rangle$*, then:*

$$s(r_a) \simeq s(r_b) \textit{ if and only if}$$
$$(LHS_a = LHS_b \land RHS_a = RHS_b) \lor (LHS_a = RHS_b \land LHS_b = RHS_a)$$

The stoichiometry definition $Stoich(r)$ (Definition 9) is insufficient to integrate reactions from distinct domains for two reasons:

- Metabolites in distinct domains have an unique identity (Definition 4), therefore the stoichiometry of reactions with metabolites in different namespaces are never equal.
- In some cases, it is necessary to exclude irrelevant compounds; the proton (H^+) is a common example of exclusion since it is added to the stoichiometry based on the protonation of the compounds to balance the reaction.

The integrated stoichiometry substitutes every single metabolite with the integrated metabolite domain function $\mathcal{I} : \mathcal{M} \mapsto \wp(\mathcal{M})$. This expands single metabolites m into metabolite sets M that map all metabolites of m that are identical (clusters).

Definition 11. *Reaction Integrated Stoichiometry*

$IntegratedStoich(r, \mathcal{I}, \mathcal{X}) = \langle LHS^{\mathcal{I}}(r, \mathcal{I}, \mathcal{X}), RHS^{\mathcal{I}}(r, \mathcal{I}, \mathcal{X}) \rangle$*, where:*

$LHS^{\mathcal{I}}(r, \mathcal{I}, \mathcal{X}) = \{(\mathcal{I}(\pi_1(p)), \pi_2(p)) \in \wp(\mathcal{M}) \times \mathbb{R} \mid p \in LHS(r) \land \pi_1(p) \notin \mathcal{X}\}$

$RHS^{\mathcal{I}}(r, \mathcal{I}, \mathcal{X}) = \{(\mathcal{I}(\pi_1(p)), \pi_2(p)) \in \wp(\mathcal{M}) \times \mathbb{R} \mid p \in RHS(r) \land \pi_1(p) \notin \mathcal{X}\}$

The $IntegratedStoich(s, \mathcal{I}, \mathcal{X})$ (Definition 11) expands the stoichiometry $Stoich(r)$ by replacing the metabolites in the tuples with metabolite sets. This allows to compare the reactions if they share the same metabolite sets in the stoichiometry.

The exclusion set $\mathcal{X}$ defines the metabolites to be excluded from the stoichiometry. For the integrated stoichiometry, $\mathcal{X}$ is replaced by $\mathcal{X}^{\mathcal{I}} = \bigcup_{x \in \mathcal{X}} \mathcal{I}(x)$ that expands the excluded metabolites to their respective integrated sets.

The reaction sets are added to the integrated space $\mathcal{I}$ (Definition 8) expanding the sets of integrated instances. The stoichiometry also gives additional information on the compounds. Let m_1, m_2 be two distinct metabolites in $\mathcal{M}$, if $\mathcal{I}(m_1) = M_1$ and $\mathcal{I}(m_2) = M_2$. If there are two reactions that have partial stoichiometry match with the exception of m_1 and m_2, this might imply that $m_1 \equiv m_2$.

Example 4. *Partial Reaction Matching*

Let $m_1, m_2 \in \mathcal{M}$, such that $m_1 \neq m_2$, and $\mathcal{I}(m_1) = M_1, \mathcal{I}(m_2) = M_2$. Let $r_1, r_2 \in \mathcal{R}$, such that $r_1 \neq r_2$.

Let $s^{\mathcal{I}}(r_1) = \langle\ ((\langle M_1, v_1\rangle, p_1), (p_2, p_3)\rangle$, and $s^{\mathcal{I}}(r_2) = \langle\ ((\langle M_2, v_2\rangle, p_1), (p_2, p_3)\rangle$.

If $v_1 = v_2$, the $s^{\mathcal{I}}(r_1) = s^{\mathcal{I}}(r_2)$ if and only if $M_1 = M_2$.

The singleton sets of partial stoichiometry generate a complementary set of integrated metabolites that are candidate merges. Sets that contain a single compound for each namespace may give strong evidence of possible duplicates.

Still it is known that it is possible for two distinct reactions to differ only in the product. As an example, KEGG reactions R10950 and R03427 differ only in one metabolite, and thus this serves only as a possibility.

The *SingletonMiss* (Algorithm 2) detects all sets of integrated stoichiometry that $Stoich(r_a) \simeq Stoich(r_b)$ if one of the metabolites in either LHS or RHS is dropped from the stoichiometry. The algorithm returns two maps: the *reactions* groups the reactions such that $Stoich(r_a) \simeq Stoich(r_b)$ is only true if one of the metabolites is dropped, the *metabolites* groups the compounds that were dropped from the stoichiometry.

The $\mathcal{I}^c$ set is the integrated set that is assembled from the metabolite sets of the *metabolites* map from the *SingletonMiss*. These are all hypotheses of unintegrated metabolites that were not merged in the previous methods with shared reaction stoichiometry (with exception to themselves).

$\mathcal{I}^c$ extends the previous reaction contribution function R (Equation 2.15) with the β_r parameter given two metabolites found within the same complement set if theres is a unique namespace for each.

$$R(a, b, \mathcal{I}^c) = \begin{cases} -\alpha_r & \exists_{e_1} \exists_{e_2} [e_1 \neq e_2 \wedge \pi_1(e_1) = \pi_1(e_2) = r \wedge \pi_2(e_1) = a \wedge \pi_2(e_1) = b] \\ \beta_r & \mathcal{I}^c(a) = \mathcal{I}^c(b) \wedge size(\mathcal{I}^c(a)) = \\ & size(\{\mathcal{N}(x) \mid x \in \mathcal{I}^c(a)\}) \\ 0 & \text{otherwise} \end{cases} \tag{2.15}$$

The reaction integration is the last step of the integration cycle. Further iterations should include curation feedback.

2.5 Implementation

The integration system is implemented through a Java$^{\text{TM}}$7 library that contains several module (Figure 2.10). The core module consists on basic and generic components and interfaces of the entire system (e.g., Metabolite, Reaction).

The biodb contains resource specific implementations of entities and data access methods (e.g., KeggCompound, BiGGMetabolite, etc).

The chemanalysis module implements all cheminformatics functions using several libraries. The commercial software Marvin allows to convert defined structures to names, but the reverse is also possible. The OPSIN library [81] allows to parse molecule names to generate InChI structures. The Chemistry Development Kit [117] (CDK) is a Java cheminformatics library that provides several parsers to interpret chemical exchange formats, but also includes the JNIInchi library to read and generate InChIKeys.

Algorithm 2 Groups integrated stoichiometry having a single unintegrated metabolite.

```
1: procedure SingletonMiss($\mathcal{R}, \mathcal{E}, \mathcal{I}, \mathcal{X}$)
   Input: $\mathcal{R}$ reactions, $\mathcal{E}$ edges, $\mathcal{I}$ integrated sets, $\mathcal{X}$ metabolite exclusion
   Output: A map $reactions : S \mapsto R$ that maps the partial stoichiometry to reaction
   sets. A map $metabolites : S \mapsto M$ that maps the partial stoichiometry to metabolite
   sets
2:     for $r \in \mathcal{R}$ do
3:         $s \leftarrow IntegratedStoich(r, \mathcal{I}, \mathcal{X})$
4:         $L \leftarrow \emptyset$                    ▷ $L$ is the left side filtering all singleton $M$
5:         $R \leftarrow \emptyset$                    ▷ $R$ is the right side filtering all singleton $M$
6:         $Z \leftarrow \emptyset$                    ▷ $Z$ is the set of singletons
7:         for $\langle M, v \rangle \in \pi_1(s)$ do          ▷ Filter left pairs
8:             if $|M| \neq 1$ then
9:                 $L \leftarrow \langle M, v \rangle$
10:            else
11:                $Z \leftarrow Z \cup M$
12:        for $\langle M, v \rangle \in \pi_2(s)$ do          ▷ Filter right pairs
13:            if $|M| \neq 1$ then
14:                $R \leftarrow \langle M, v \rangle$
15:            else
16:                $Z \leftarrow Z \cup M$
17:        $s' \leftarrow \langle L, R \rangle$
18:        if $|Z| = 1$ then
19:            $reactions[s'] = reactions[s'] \cup r$
20:            $metabolites[s'] = metabolites[s'] \cup Z$
21:    return $reactions, metabolites$
```

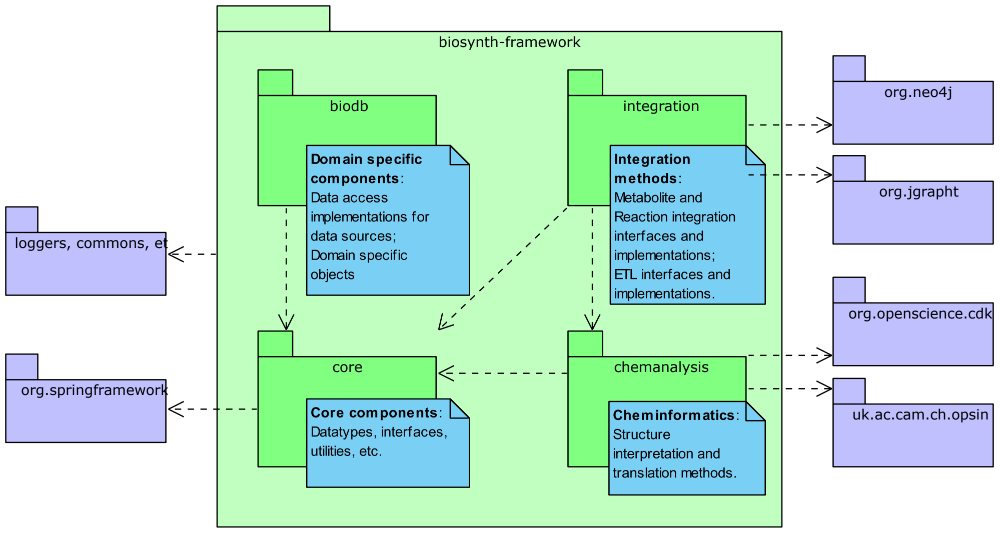

biosynth-framework
biodb
Domain specific components: Data access implementations for data sources; Domain specific objects
integration
Integration methods: Metabolite and Reaction integration interfaces and implementations; ETL interfaces and implementations.
core
Core components: Datatypes, interfaces, utilities, etc.
chemanalysis
Cheminformatics: Structure interpretation and translation methods.
loggers, commons, et
org.springframework
org.neo4j
org.jgrapht
org.openscience.cdk
uk.ac.cam.ch.opsin

Figure 2.10: The pipeline and methods are implemented as a Java library (biosynth-framework) with several modules.

The integration module provides the interfaces and methods for the integration pipeline, here the CDS access methods are also implemented while it is designed to be independent of the biodb.

2.5.1 Graph Database Systems

The flexibility of the database is one of the most important requirements of the CDS system. Biological entities and their functions are not trivially characterized. It is common that databases have a set of records with lack of literature evidence or just to simply to support an hypothesis, which in turn makes the data highly volatile and subject to future changes.

Recently, graph databases gained more attention in data integration approaches. Since graphs are common strategies to define relationships, the graph storage model suits the needs for the data complex heterogeneous models allowing for easy *ad hoc* addition of new relations [119].

In recent years, there is a increasing popularity of using graph databases for bioinformatics application, since they have proven to be easier to maintain, while having much higher speed up gains for extensively relational queries [56, 70, 49].

The neo4j graph database is used to build the CDS. Like most of the NoSQL databases, neo4j offers a flexible schema. The addition of new entities and attributes can be easily achieved with little impact in the database performance, allowing an incremental schema design to adapt the needs of the data.

Each record in neo4j databases is either a node (vertex) or relationship (edge). Both nodes and relationships are allowed to have a set of arbitrary properties, while nodes are allowed to have several `Label`s and relationships may only be assigned to a single `RelationshipType`. In order to organize the database information, a set of rules is applied to define a simple generic data schema to fit all the data requirements.

Each neo4j Node and Relationship must have a unique numeric identifier, being the *id*s of the objects stored in the properties of the node as a string attribute.

To define the namespace of the object, a Label is assigned to the Node, and in its properties the namespace string attribute marks the Label that corresponds to the object namespace (since Nodes are allowed to have several Labels).

The central data storage defines a relationship and class hierarchy for the domain of metabolites and reactions (Figure 2.11). Since each node may have several labels, the abstract labels are used to define the class of the objects. These are the supersets of $\mathcal{M}$, $\mathcal{R}$, $\mathcal{P}_m$, $\mathcal{P}_r$ that are labeled as $Metabolite$, $Reaction$, $MetaboliteProperty$ and $ReactionProperty$, respectively.

A vocabulary is defined to represent the interactions between the instances. These are the meanings given by the $e \in \mathcal{E}$. As an example, given $e = (m, p)$, such that $m \in \mathcal{M}$ and $\mathcal{N}(p) = InChI$, the relationship e is limited to *has_inchi*.

The Cypher query language is a neo4j specific query language to manipulate the database. For the uniqueness constraint, the following query allows to force the database to refuse duplicate attributes within a Label (n):

$$\forall_{n \in \mathcal{N}} \texttt{ CREATE CONSTRAINT ON (o:}n\texttt{) ASSERT o.id IS UNIQUE}$$

Relational queries can be easily achieved using the following cypher expression:

$$\texttt{MATCH (}\varphi_1\texttt{)-[}\psi_1\texttt{]->(}\varphi_2\texttt{) RETURN (}\tau\texttt{)}$$

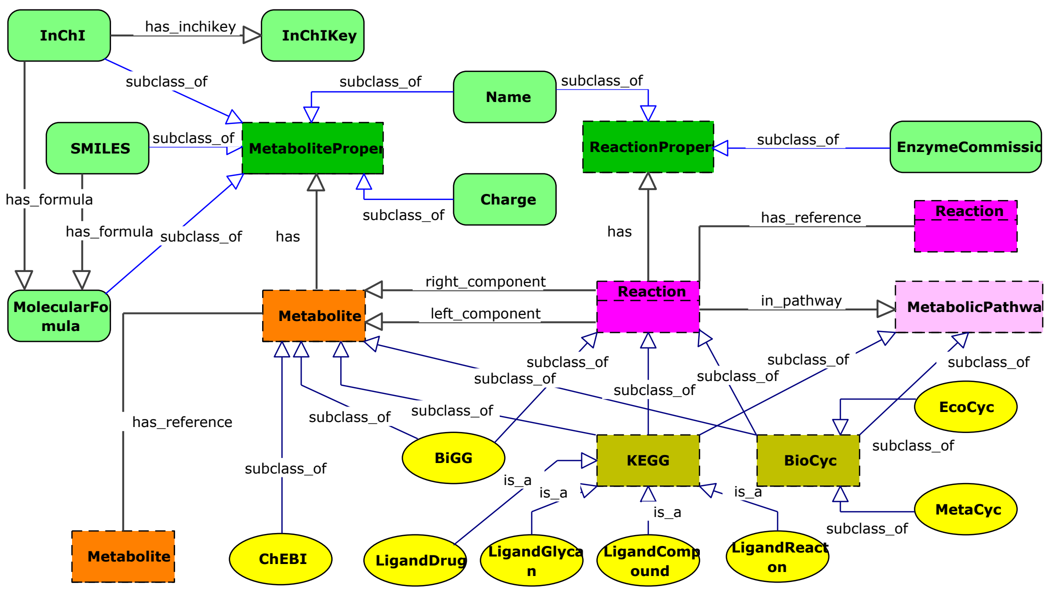


Figure 2.11: Neo4j Labels defined for the CDS namespace domain. Metabolite, Reaction, MetaboliteProperty, ReactionProperty represents the sets $\mathcal{M}, \mathcal{R}, \mathcal{P}_m$ and $\mathcal{P}_r$, respectively. Light Green - Property namespaces. Yellow - Database namespaces. Additional namespaces can be added if needed (e.g., Metabolic Pathway).

The φ_1 and φ_2 are nodes matching predicates with the following pattern:

$$\varphi = \texttt{variable:}Label_1\texttt{:}\ldots\texttt{:}Label_x\ \{attribute_1\texttt{:}value_1\texttt{,}\ \ldots\ \texttt{,}attribute_y\texttt{:}value_y\}$$

allowing to filter nodes by label and attributes. As an example, to select a particular set of metabolites of a database, the following expression would be sufficient:

```
MATCH (m:LigandCompound) RETURN m
```

or to select a particular compound:

```
MATCH (m:LigandCompound {id:"C00022"}) RETURN m
```

The omitted arrow block is the traversal condition that matches if connected to the node predicate φ_2 and all the connections satisfy the predicate ψ_1. As an example, the following expression catches every compound in the system that have a particular chemical formula:

```
MATCH (m:Metabolite)-[:has_formula]->(m:MolecularFormula {id:"C3H4O3"})
RETURN (m)
```

In this case, this would allow matching graph paths of length 1 (although the result could be a star-shaped subgraph the maximum diameter would be 1), the expression can be extended with the addition of arrow blocks $\ldots$-[ψ_2]$\rightarrow$(φ_3)-$\ldots$-[ψ_j]$\rightarrow$(φ_{j+1}).

Alternatively, the reaction predicate accepts a multiplicity value:

```
MATCH (m:LigandCompound {id:  "C00022"})-[:has_reference*]-(n:Metabolite) RETURN (n)
MATCH (m:LigandCompound {id:  "C00022"})-[:has_reference*..3]-(n:Metabolite) RETURN (n)
```

The first expression catches the entire subgraph that connects the KEGG Compound C00022 with other metabolites by cross-referencing, while the second limits the search up to a diameter of 3.

The integration exploits the capability of the Cypher query language to perform all the necessary queries, to implement the proposed integration methods.

2.5.2 Extract and Translation Methods

The implementation of extraction methods is considered the main bottleneck for the expansion of the integrated metabolic database. Third-party resources provide their own data access methods and protocols to expose their data, while most of the modern platforms implement a REST API, but the output format and schema may be different for each of the external resources.

The addition of a new database to the system implies the implementation of a specific `DataAccessObject` (DAO) and a `Transform` function (Figure 2.12). These are the only components that are required per data source.

The DAO is responsible for listing and extracting database specific objects e from third-party resources. The entities extracted from the DAO do not need to conform the standards of the CDS universe, while its main purpose is to interpret and extract correctly the desired attributes.

The transform function $T : e \mapsto \mathcal{G}$ conforms the raw entities into the respective graph objects o and edges e. The transform function is responsible for giving the namespace of each o. A dictionary to conform referenced databases is needed to unify the different names given by the databases (e.g., KEGG-CPD, KEGG Compoumd, Ligand Compound).

The ETL pipeline is designed to only integrate the relationships of the loaded instances, while integration of the metabolites and reactions is done afterwards.

The cleansing is only performed after the transformation (Figure 2.13) and it is responsible to fix the properties of the transformed instances. The need to implement a cleansing system is to perform minimal standardization for the properties to avoid proliferation of equivalent attributes. As an example, the atom order of the chemical formulas can be rearranged in several ways (e.g., $H_2O, OH_2, H_2O_1, O_1H_2$). In fact, all the properties could be loaded as they are, then the cheminformatics subsystem could unify all the properties in the system, but many of these occurrences would be trivial.

The formulas are rewritten to a standard atom order using the CDK library. The `MolecularFormulaManipulator` is used to rewrite the formulas with the `getMajorIsotope-`

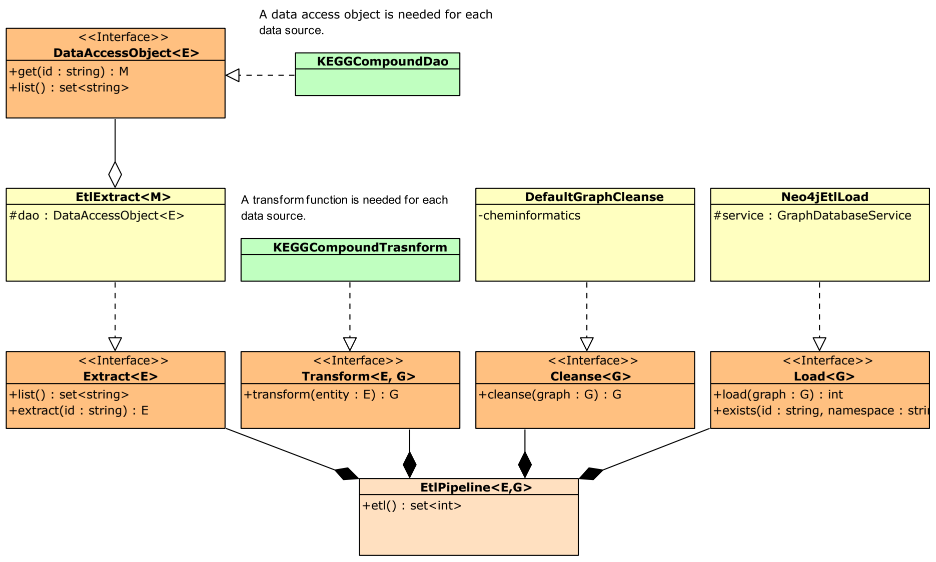


Figure 2.12: The ETL pipeline requires a DAO and Transform function for every data source. Databases that contain both metabolites and reactions count as two distinct data sources.

`MolecularFormula` function. The names are also modified to lower case to minimize the noise in the central database, but for all modifications, the original value is stored in the attributes of the e and flagged as corrected.

The current library implements DAO and Transform objects for several pathway and metabolite databases. The details era given as follows:

KEGG: Kyoto Encyclopedia of Genes and Genomes The KEGG database has a REST API to fetch its data. The only format given is a semi structured text file (Figure 2.14).

A parser was developed to split the content from the text file and to extract the attributes of each record. Without a defined schema, it is not trivial to detect the existing attributes of all the records and their possible data-types or missing values.

The API exposes a simple REST access:

```
http://rest.kegg.jp/<operation>/<argument>[/<argument2[/<argument3> ...]]
```

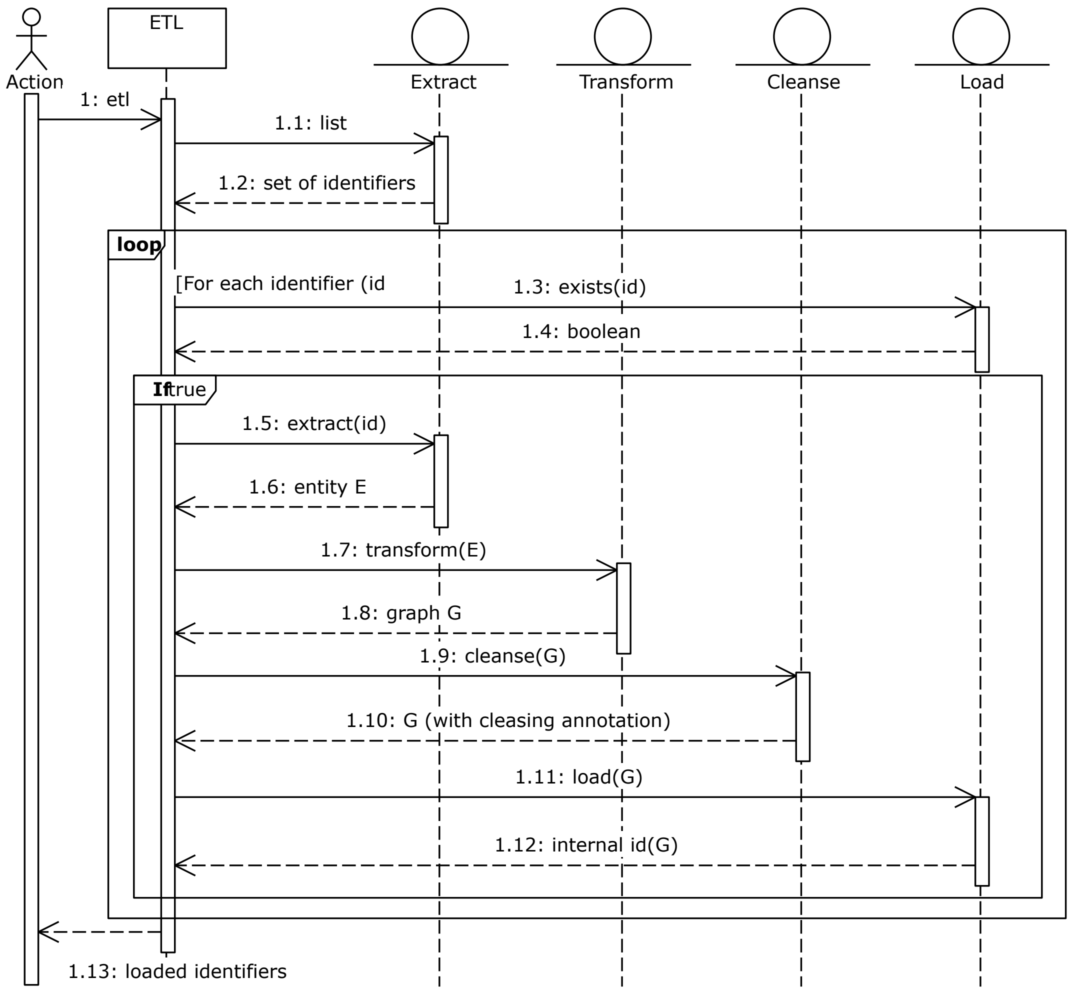


Figure 2.13: The ETL pipeline. The extract component returns all identifiers of a third-party database and loops for each the ETL process.

For extraction, only the `list` and `get` operations are needed.

The KEGG database is split into three sets, corresponding to compounds, glycans and drugs. Each of these records have a specific starting prefix capital letter, C, G, and D respectively, and R for reactions. Each database is stored in a different namespace.

MetaCyc: Metabolic Pathway Database The BioCyc [65] collection covers several databases known as pathway/genome databases (PGDBs). The MetaCyc [16] PGDB is the reference database of Biocyc covering the metabolism of the other organism specific PGDBs (e.g., EcoCyc, YeastCyc, HumanCyc).

The PGDBs are created and managed by the Pathway Tools [66] platform that provides a query API to retrieve and search data in a XML format. The BioVelo [77] query processor is a specific query language to allow complex queries over the Pathway Tools relationship schema, available in:

```
https://websvc.biocyc.org/xmlquery?[x:x<-[PGDB]^^[CLASS]]
```

The individual objects are obtained from the `xmlget` function:

```
https://websvc.biocyc.org/getxml?[PGDB]:[OBJECT-ID]
```

For the MetaCyc database, the `PGDB` is assigned to `META` but the access to other databases uses the same interface just by changing the `PGDB` (e.g., `ECOLI` - *Escherichia coli*, `YEAST` - *Saccharomyces cerevisiae*, `HUMAN` - *Homo sapiens*),

BiGG: Biochemical Genetic and Genomic knowledgebase The BiGG database is split into the old BiGG (BiGG1) [110] database[4] and the latest BiGG [68] database[5]. Unlike other genome pathway databases, the BiGG databases are built based on published genome-scale metabolic models. The early BiGG1 database was built by merging 10 models, while the recent BiGG contains more than 80 models with standardized identifiers and a public API.

[4]http://bigg1.ucsd.edu/

[5]http://bigg.ucsd.edu/

```
ENTRY        C06142                       Compound
NAME         1-Butanol;
             n-Butanol;
             Butan-1-ol
FORMULA      C4H10O
EXACT_MASS   74.0732
MOL_WEIGHT   74.1216
REMARK       Same as: D03200
REACTION     R03544 R03545 R11343 R11344 R11448
PATHWAY      map00650  Butanoate metabolism
             map01120  Microbial metabolism in diverse environments
             map01220  Degradation of aromatic compounds
ENZYME       1.1.1.-          1.1.2.9          1.1.5.11         1.14.13.230
BRITE        Pharmaceutical additives in Japan [BR:br08316]
              Solubilization agent
               D03200  [101088] Butanol
              Solvent
               D03200  [101088] Butanol
DBLINKS      PubChem: 8398
             ChEBI: 28885
             ChEMBL: CHEMBL14245
             KNApSAcK: C00035814
             PDB-CCD: 1BO
             3DMET: B00907
             NIKKAJI: J2.374D
ATOM         5
             1    C1b C     24.4358   -15.4702
             2    C1b C     23.2158   -14.7756
             3    C1b C     25.6384   -14.7639
             4    C1a C     22.0074   -15.4761
             5    O1a O     26.8526   -15.4702
BOND         4
1      1   2 1
2      1   3 1
3      2   4 1
4      3   5 1
///
```

Figure 2.14: KEGG Compound - C06142 (n-butanol), an example of the KEGG text format. Attributes are separated by fixed space columns.

abbrv	name	formula	c	cmp	kegg_id	cas_id	bigg_id	found in
glu-D	D-Glutamate	C5H8NO4	-1	[c]	C00217	6893-26-1	34285	1,10,3,4,5,7
pre6b	Precorrin 6B	C44H49N4O16	-7	[c]	C06319		1800230	10,4
23dhb	2,3-Dihydroxybenzoate	C7H5O4	-1	[c]	C00196	303-38-8	34227	1,5
ru5p-L	L-Ribulose 5-phosphate	C5H9O8P	-2	[c]	C01101		36809	1,10,5,7
18harachd	18 hydroxy arachidonic acid	C20H31O3	-1	[r]			2300206	2
ca2	Calcium	Ca	2	[e], [c], [p]	C00076	7440-70-2	33764	10,2,5,9
lpp	lipoprotein	XC16H30O1	0	[p]	C01834		2707630	5

Figure 2.15: Example of BiGG1 CSV records. abbrv - abbreviation, c - charge, cmp - compartment, found in - 1) E. coli iJR904; 2) H. sapiens Recon 1; 3) H. pylori iIT341; 4) P. putida iJN746; 5) E. coli iAF1260; 6) S. cerevisiae iND750; 7) S. aureus iSB619; 8) E. coli textbook; 9) M. barkeri iAF692; 10) M. tuberculosis iNJ661;

The BiGG1 database allows to download the entire database to a file with the CSV format (Figure 2.15). However, a profiling step is still required to determine the data-type of each column. Some columns may be defined as single values (strings, integers, decimals) or lists of entities (e.g., compartment, found in).

The latest BiGG provides a documented REST API to browse and fetch data in the JSON format. The list of all metabolites and reactions can be accessed through REST web calls:

```
http://bigg.ucsd.edu/api/v2/universal/reactions
http://bigg.ucsd.edu/api/v2/universal/metabolites
```

HMDB: The Human Metabolome Database Some databases are oriented to a specific species. In the case of the HMDB [124], it is dedicated to metabolites found in the human body. The HMDB is contains only metabolite information, but with an enriched set of features such as, sprectroscopic, quantitative, analytic and physiological information about the human metabolome.

HMDB allows to bulk download the entire dataset in XML format from their web page.

LIPID Metabolites and Pathways Strategy The LIPID MAPS database is dedicated to a specific class of metabolites which are the lipids. The purpose of this database is to provide a better classification system for the lipids, while focusing also in the chemical structure of these and the ability to draw such complex structures.

Lipids usually contain large chains that are troublesome to draw and display using current methods to generate atom coordinates, such that for the same lipid it is common to be displayed in many different conformations in distinct resources.

The LIPID MAPS database can be bulk downloaded in the SDF format.

Model SEED The ModelSEED [7] database was developed to connect biochemical reactions to genome annotation for the purpose of model reconstruction. The reactions in the ModelSEED database are assigned to particular gene roles, which are given by the RAST annotation systems, allowing to directly link the metabolic function from the genome annotation.

The metabolic information of the ModelSEED database can be found in their github repository[6] in both TSV and JSON formats.

2.6 Results

2.6.1 Building an integrated metabolite database

Here, an integrated metabolic database is generated using the implemented pipeline described in the previous sections. Each step of the pipeline is detailed with a discussion in the following sections.

The implemented ETL modules are ran to load several external resources into the CDS (Table 2.3). At the time of writing, most of the databases collected match the latest versions available.

It is possible to observe that there is a significant number of proxy instances in the CDS after the ETL step. This may imply that there is a certain lack of synchronization between resources, since a proxy is usually caused by database referencing to a non existing entity

[6]https://github.com/ModelSEED/ModelSEEDDatabase

Table 2.3: Biochemical resources in the final integrated database

Resource	Version	Type	Format	Instances	Proxies
KEGG Compounds	84.0 (October 1, 2017)	REST	text	18.111	443
KEGG Glycans	84.0 (October 1, 2017)	REST	text	11.015	2
KEGG Drugs	84.0 (October 1, 2017)	REST	text	10.440	0
MetaCyc	21.1 (August 15, 2017)	REST	xml	18.242	1343
BiGG1	2010 (discontinued)	dump	CSV	2.835	255
BiGG	1.4 (October 14, 2017)	REST	json	6.215	34
ModelSEED	190fb3e (master)	GitHub	json	27.693	0
LIPID MAPS	6Dez16	dump	SDF	40.772	10
HMDB	3.6 (2014)	dump	json	41.758	11

in another resource. In some cases, this could also be related to bad referencing (e.g., malformed identifiers in the source databases).

Most of the BiGG1 proxies have origin on the ModelSEED instances. A closer analysis reveals that these are references to external metabolic models that were not included in BiGG1. However, in the ModelSEED they were assigned as BiGG1 metabolites.

The majority of MetaCyc proxies came from ModelSeed (around 1000), but also both HMDB and BiGG have referenced around 100-200 deleted records. Databases that have a higher number of proxies are also more popular, since for both KEGG and MetaCyc they are subject to regular updates, and other databases are unable to keep up with the regular update cycles.

Regarding the integration logic, the proxy records do not take part in the integration methods since they contain no information except the origin of the reference.

A summary of raw attributes (Table 2.4) allows the identification of type of attributes found in the raw instances. Of all the attributes, the name and formula have regular presence in most of the collected instances, with the exception for the KEGG Glycan database, which does not provide any molecular formulas (instead it provides the sugar composition). Many other compound characteristics are not taken into account for the integration methods. These attributes can be inherited in the final database since they are domain specific.

To evaluate the designed integration methods an instantiation of the parameters is defined ignoring manual curation (Table 2.5). The curated compound sets are used later to compare against the predicted integration, the databases used for integration are the KEGG (compounds only), MetaCyc, and both the BiGG and BiGG1 databases. These are the main metabolic pathway databases and they contain both reactions and metabolites. Since the BiGG database uses references from MetaNetX, these are removed from the system. The BiGG1 references were kept.

The integrated reference space of MNX is used to compare against the results from the implemented methods in this work.

As of today, there are three versions of the MNX (version 1.0, 2.0, 3.0), but only the last version is used (Figure 2.16). The MNX integration suffered many modifications from version to version, and it is possible to see a significant increase of the total of instances, but also in the latest version many discontinued databases were discarded from the integration.

In fact, the MNX contains many databases not studied in this work. Only the compound instances that were found within our metabolite domain ($\mathcal{M}$) are considered. The instances of other databases are removed from the MNX set.

The latest version of MNX discarded the first version of BiGG (BiGG1), for comparison purposes the BiGG1 database is discarded from the predicted sets.

For integration evaluation purposes, a set of curated integrated metabolites was manually annotated to compare against the automated integration. The set consists on the integration of KEGG Compound, MetaCyc, BiGG1 and BiGG, covering a total of 782, 785, 761, 761 metabolites, for each of the databases respectively.

All the curated metabolites participate in at least one reaction, since these are the most relevant compounds for the integration.

Table 2.4: Frequency of the raw attributes collected from each of the resources. MS - ModelSEED, MC - MetaCyc, LC - KEGG Compound, LG - KEGG Glycan, LD - KEGG Drug, HM - HMDB, B - BiGG, B1 - BiGG1

Attribute	MS	MC	LM	LC	LG	LD	HM	B	B1	Type	Mapping
name	1.00	1.00	0.78	1.00	0.12	1.00	0.99	1.00	1.00	String	Name
synonyms			0.22							String	Name
systematicName			0.87							String	Name
iupacName							0.99			String	Name
abbreviation	1.00									String	
formula	0.82	0.71	0.99	0.96		0.82	0.99	0.93	1.00	String	Formula
composition					1.00					String	
inchi		0.70	0.99				0.99			String	InChI
inchiKey			0.99							String	InChIKey
inchikey							0.99			String	InChIKey
smiles		0.86					0.99			String	SMILES
structure	0.78									String	InChI
charge		0.86							1.00	Number	
defaultCharge	1.00									Number	
mass				0.88	0.94					Number	
exactMass			0.99							Number	
cmlMolWeight		0.72								Number	
molWeight		0.72								Number	
averageMolecularWeight							0.99			Number	
monisotopicMoleculateWeight							0.99			Number	
deltaG	0.97									Number	
deltaGErr	0.97									Number	
gibbs		0.70								Number	
description		1.00								String	
oldIdentifiers								1.00		String	
internalId									1.00	Number	
tissues							0.15			String	
ontology_origins							0.89			String	
ontology_status							0.99			String	
ontology_biofuncions							0.79			String	
ontology_applications							0.85			String	
ontology_cellular_locations							0.94			String	
biofluids							0.13			String	
secondary_accession							0.13			String	
pubchemSubstanceUrl			0.92							String	
lipidMapsCmpdUrl			1.00							String	
category			1.00							String	
mainClass			1.00							String	
subSlass			0.92							String	
classLevel4			0.26							String	
core	1.00									Boolean	
cofactor	1.00									Boolean	
obsolete	1.00									Boolean	
remark				0.19	0.18	0.68				String	
comment				0.86	0.22	0.54				String	
status			1.00							Boolean	

Table 2.5: Parameters values defined for the similarity function.

Function	Parameter	Type	Value
N	α_n	similarty	0.3
	β_n	similarty	0.15
	γ_n	similarty	0.1
	κ_n	penalty	1
	ρ_n	names count	3
S	α_i^+	similarty	1
	β_i^+	similarty	0.8
	α_i^-	penalty	1
	β_i^-	penalty	1
	α_s^+, β_s^+	similarty	0
	α_s^-, β_s^-	penalty	0
X	α_x	similarty	1
	β_x	similarty	0.5
F	α_f	similarty	1
	β_f	similarty	1
	γ_f	similarty	0.8
	η_f	similarty	0.8
C	α_c	similarty	0
	β_c	penalty	0
R	α_r	penalty	10
	β_r	similarty	1

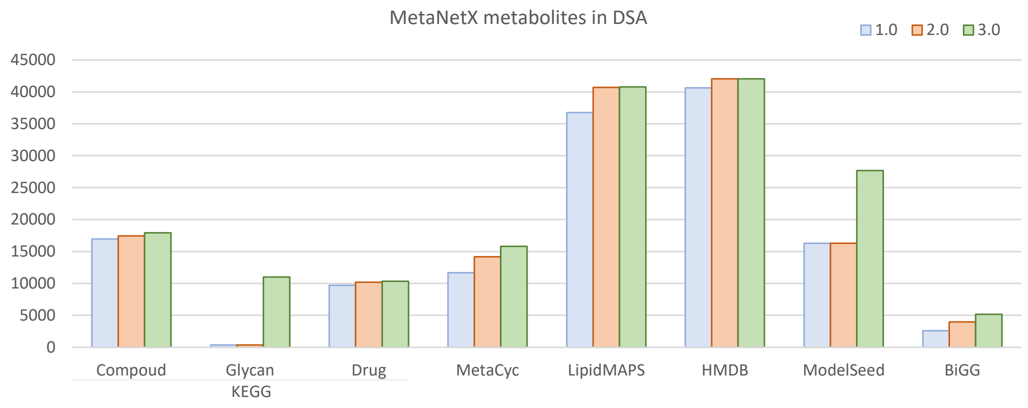


Figure 2.16: Metabolite instances that are both present in the CDS and each of the MNX versions.

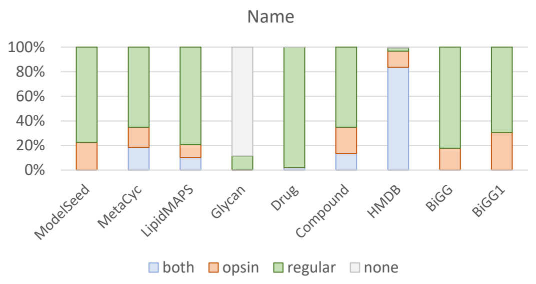
2.6.2 Statistics for the knowledge expansion

To assess the impact of the cheminformatics module, the properties are split into two categories (Figure 2.17): the source properties are all properties inherited from the data sources, while the translated properties are the generated from other properties.

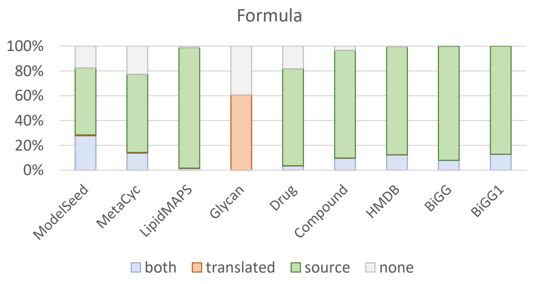
The MetaCyc and HMDB were the only databases that had high present of a both translated and source structures for a single compound (Figure 2.17c and 2.17d). The KEGG structures were all translated from the Mol files with the exception of glycans that provided no structural information. All structures in the BiGG database were generated, since it does not provide any original information regarding these.

Only the KEGG Glycan took additional benefit from formula translation (Figure 2.17b), but it was generated from the KEGG glycan composition and it is KEGG specific. The presence of both translated and source formulas are also low compared to the structures, which may indicate low structural conflict but possible stereo or protanation conflicts.

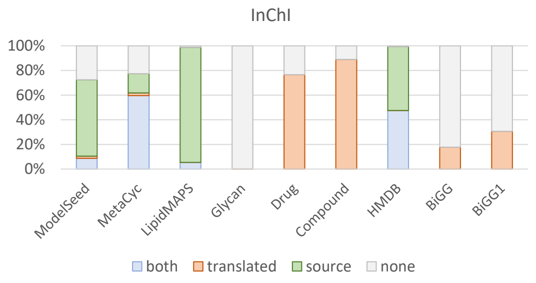
Name to structure (Figure 2.17a) displays if the metabolites contain names that were parseable by the OPSIN library. Only HMDB had a majority of the compounds with at least one synonym that is compatible with the OPSIN library, the remaining databases all scored around 20% and 30%. This is due the fact that HMDB had a specific field for a IUPAC name that was generated from the ChemAxon tool (Table 2.4).



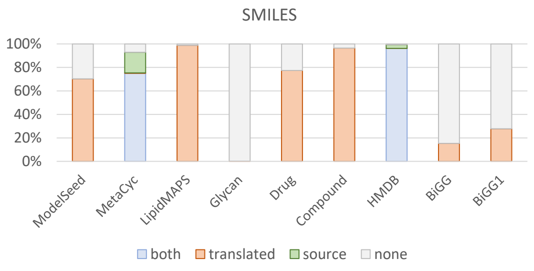
(a) Percentage of instances with OPSIN and/or regular names.



(b) Percentage of instances with formulas.



(c) Percentage of instances with InChI.



(d) Percentage of instances with SMILEs.

Figure 2.17: Property coverage of the database instances. Translated (orange) properties were generated by the cheminformatics module. IUPAC names are all compounds that had at least one name that was converted to InChI by OPSIN.

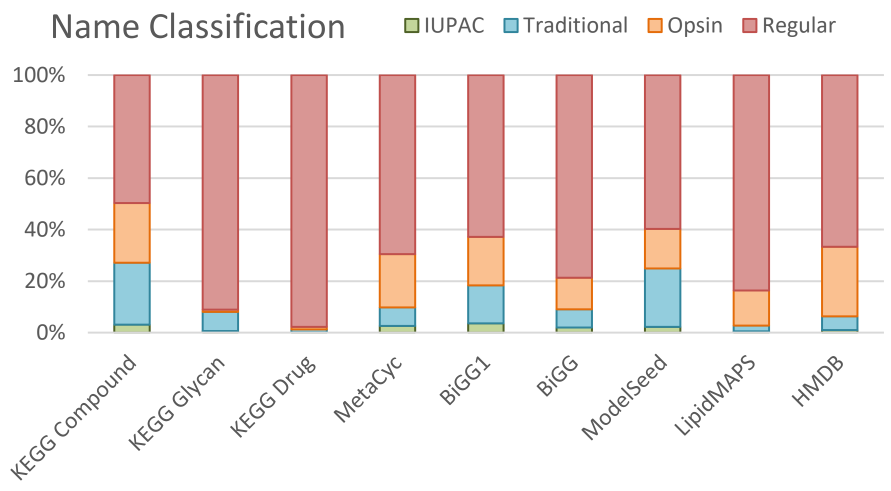
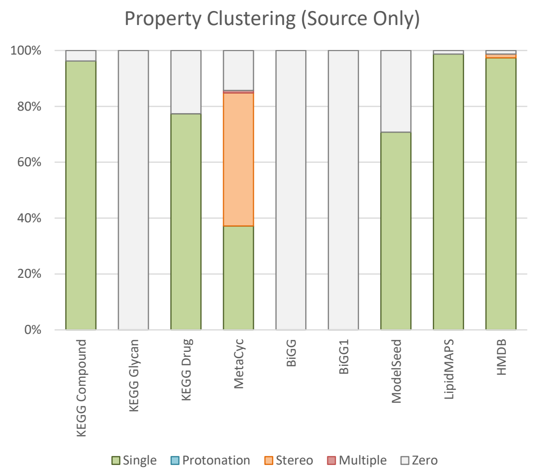


Figure 2.18: Classification of each name property in the databases. IUPAC and traditional names that had a match with a name generated with the Marvin *StructureToName* plugin. OPSIN are names that are parseable with the OPSIN library. Regular are all names that fail all the previous tests.

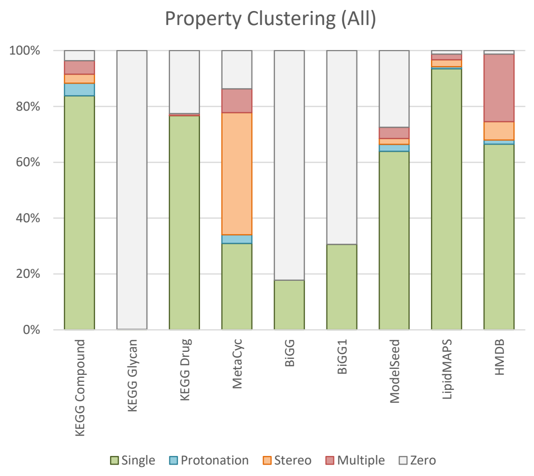
To evaluate structure to name translation (Figure 2.18), the existing name properties are flagged as IUPAC, Traditional, OPSIN or regular, receptively. The top priority are the IUPAC names that were generated from all the structures in the CDS using the Marvin *StructureToName* plugin, then matched with the existing names in the CDS. Only a few names did match to the IUPAC names generated from the structures. However, if using the traditional name given by the plugin, these numbers are much higher. The OPSIN library is able to cover many more names compared with the ones generated from the Marvin plugin. Perhaps the lack of adoption of IUPAC names is because of the non human friendly nomenclature, which in many cases leads to very large names with many repeating sequences.

In many cases, the translated properties may add an additional structure to molecules because of inconsistencies in the existing ones. This, however, is only a problem if the additional structure and the original are not equivalent molecules (i.e., same molecule different protonation state).

To find structural conflicts, the properties are clustered together since the translation



(a) Using only source properties.



(b) Using all properties.

Figure 2.19: Classification of the property clusters (SMILES, InChI, Mol) for each database. Single - all structures match. Protonation - multiple structure, but same molecule. Stereo - multiple structure, but same connectivity, Multiple - multiple structures with different connectivity.

method connects properties to each other (Figure 2.7). For each metabolite, three types of clusters are evaluated: single structure are all metabolites that connect to structures that represent exactly one metabolite, protonation structures are metabolites that connect to several structures but represent one molecule, while stereo clusters are those with several structures differing in the stereo layer but with equal connectivity, and multiple are the molecules that differ in both stereo and connectivity.

The clustering of structures without any translated properties (Figure 2.19a) shows many cases of multiple stereo clusters in the MetaCyc compounds. The MetaCyc provides SMILES without any stereo information, this explains the high amount of both translated and source InChI and SMILES for each compound (Figure 2.17c and 2.17d). The clustering of structures with the translated properties (Figure 2.19b) shows that for some compounds there were conflicts between the generated structures and the original ones, but the numbers are low with the exception of the HMDB that had more than 20% of increase in the number of conflicts.

The *OPSIN* library was able to distinguish protonation state from the compound names. As an example, the KEGG pyruvate[7] molecule has five names displayed: *Pyruvate*, *Pyruvic acid*, *2-Oxopropanoate*, *2-Oxopropanoic acid*, *Pyroracemic acid*; only the last name was unable to parse. This is an example that the expansion module would generate a protonation conflict.

Another example is the KEGG Inosine 5'-monophosphate[8] that has 9 names assigned, unlike the previous example, 3 names were non OPSIN (*IMP, 5'-Inosine monophosphate, 5'-IMP*), while other 5 generated the correct InChI structure that matches with the Mol file from KEGG. However, one of the synonyms (*5'-Inosinate*) generated a structure without the phosphate group, since it did not make any reference to the phosphate.

To fix mismatching generated structures, every chemical structure that is generated from the cheminformatics module is excluded if it does not match the original structures. The owner of the mismatching structure is also flagged as ambiguous, this may help future decisions since these properties may be problematic for the integration methods, and should be subject to curation analysis.

2.6.3 Integration Results

2.6.3.1 Comparison with MNX

The integration cycle was ran automatically five times, each cycle feeding the next with the reaction context. This generates five sets that are represented as $i_0, i_1, \ldots, i_4$.

The first analysis is to check the sizes of each integrated compound set. In the final integration (iteration i_4) the number of integrated sets of size 2 and 3 are of a similar total amount compared to the sets provided from the MNX integration (Table 2.6). The usage of negative rules allowed for a more conservative approach for merging since a single matching property (e.g, a cross-reference) may not be sufficient.

The reactions have proven to be essential to catch up with the numbers of the MNX

[7]KEGG Compound: C00022 - http://identifiers.org/kegg.compound/C00022
[8]KEGG Compound: C00130 - http://identifiers.org/kegg.compound/C00130

Table 2.6: Cluster sizes of MetaNetX and the integrated database of KEGG (Compound), MetaCyc, BiGG.

$\mathcal{I}$	2	3	4	5	6+
MNX					
MNX 1.0	4984	1248	134	34	13
MNX 2.0	5284	1564	200	60	31
MNX 3.0	5832	1358	191	55	34
Integration Iterations					
i_0	4962	1253	84	23	34
i_1	5472	1300	87	24	
i_2	5584	1309	89		
i_3	5624		90		
i_4	5637				

integration. However, after three iterations the gains from reactions have dropped to insignificant levels per iteration.

Since the methods applied are more conservative, the number of oversize clusters is lower. The ideal scenario are clusters of size 3 and for both of the integrations they found approximately the same amount of sets.

Perhaps the total number of 1000 integrated compounds is enough to describe most of the metabolite products and intermediates for essential pathways in cell metabolism, to cover DNA, RNA, protein, fatty acids and cofactors biosynthesis in prokaryotes (since most of the BiGG content is about prokaryotic pathways).

2.6.3.2 Curated Set

The sets translation function (Algorithm 3) maps elements of an integrated set against another set. To evaluate the integrations, each set is mapped to (Translate($\mathcal{I}_{cura}, \mathcal{I}$)) and from (Translate($\mathcal{I}_{cura}, \mathcal{I}$)) the curated set.

The assumption is that elements in the curated set that are merged together are considered as true positives, and elements that are in the curated set but not together are considered true negatives, and for the remaining elements that are not present, then nothing is known about these.

If the curated set is bidirectionally mapped against the integration, then:

- Match: occurrences of C mapping to a single M and vice versa.
- Mismatch: occurrences of M matching multiple C sets. These are examples of over-integration (splitting is necessary).
- Miss: occurrences when C matches to empty set or with multiple M sets. These are examples of under-integration (merging is necessary).
- No information: occurrences of M matching to empty set. These are examples when no information is found in the curation sets regarding to M.

Algorithm 3 Translates integrated metabolite sets of two integrated domains

```
1: procedure TRANSLATE($\mathcal{I}_a$, $\mathcal{I}_b$)
   Input: $\mathcal{I}_a$ integration $a$, $\mathcal{I}_b$ integration $b$
   Output: $T$ mapping of sets of $\mathcal{I}_a$ to sets of $\mathcal{I}_b$
2:     $T = \{\}$
3:     for $M \in \mathcal{I}_a$ do
4:         for $m \in M$ do
5:             $T[M] \leftarrow T[M] \cup \mathcal{I}_b(m)$
6:     return $T$
```

Example 5. *Sets Translation*

Let, $A = \{a_1 = (1,2), a_2 = (3,4), a_3 = (5,6), a_4 = (7,8)\}$ *and* $B = \{b_1 = (4,9,10), b_2 = (7,11), b_3 = (5,6), b_4 = (8,12)\}$

$T(a_1) = (\emptyset)$, $T(a_2) = (\emptyset, b_1)$, $T(a_3) = (b_3)$, $T(a_4) = (b_2, b_4)$

The sensitivity (true positives) and specificity (true negatives) of the integration would be how many instances were correctly merged or split.

Table 2.7: Translation cases between the curated sets and the integration. C is a set in $\mathcal{I}_{cura}$. M is a set in $\mathcal{I}$. a) - Match; b) - Mismatch; c) - Miss; d) No information.

Scenario	Class	1.0	2.0	3.0	i_0	i_4
$T = \text{Translate}(\mathcal{I}_{cura}, \mathcal{I})$						
$T[C] = \emptyset$	c)	29	26	36	40	19
$T[C] = (\emptyset, M)$	a), c)	144	81	102	120	124
$T[C] = M$	a)	619	690	655	635	643
$T[C] = (M_0, M_1, \ldots, M_i)$	c)	20	15	19	17	26
$T = \text{Translate}(\mathcal{I}, \mathcal{I}_{cura})$						
$T[M] = \emptyset$	d)	5616	6344	6682	5571	6293
$T[M] = (\emptyset, C)$	d), a)	123	158	160	118	131
$T[M] = C$	a)	668	631	621	663	680
$T[M] = (C_0, C_1, \ldots, C_i)$	b)	6	6	7	4	4

Table 2.8: Confusion matrices of the MNX and i_4 integrations compared against the curated sets.

MNX

		Actual	
		$\equiv$	$\not\equiv$
Predicted	$\equiv$	1915	57
	$\not\equiv$	218	2579938

i_4

		Actual	
		$\equiv$	$\not\equiv$
Predicted	$\equiv$	1889	28
	$\not\equiv$	244	2579967

The first test is to evaluate the MetaNetX integrated sets to test the performance of the curated sets.

As an exception, the curated set splits the protonation set of a few compounds, namely between H_2O and OH, NH_3 and NH_4, H_2CO_3, HCO_3 and CO_3. All of these three sets were found to be merged in all of the three version of MetaNetX references.

From the curated set it is possible to compute the amount of true positives and true negatives edges such that $\phi(e) > \omega$. These are compared with the predicted clusters of this study and with the MNX 3.0 integration. As expected the amount of true negatives outnumbers the true positives (around 0.01% of the $e \in \Gamma$ are true positives).

Given the selected configuration, our method resulted in less 50% of false positives (29 occurrences), while regarding false negatives there was an increase of 26 cases (Table 2.8).

In fact both numbers are similar, but it is hard to tell how exactly the MNX clusters were integrated and if there was any post integration curation.

2.7 Conclusions

The bottleneck to cover more data sources is limited to how much effort is spent to develop the ETL interface. The understanding of the data schema, but also the data access mechanism of these databases is one of the most laborious tasks.

A quality integration is achievable, but errors are unavoidable. Having a more conservative approach allows to generate more confidence in the clustering but in return more instances are left unintegrated. Therefore, manual curation of data is necessary for a proper consensus database, since it is unpractical to verify entire databases, this task demands community curation efforts.

The integrated pipeline proven to be doable for on-demand integration of metabolites and reactions fitting the needs for the user of how conservative should the final dataset be, thus allowing to control the quality of the integration versus number of integrated instances.

The curation function is a powerful method that allows to combine solutions of previous integrations but also external sets, allowing to include community curation efforts and other sources of integrated databases.

The implemented data management system is designed to be flexible to the addition of further domains of instances. By using the neo4j graph database it is possible to include additional metabolic domains without having any negative impact with the previous entities and methods, while taking immediate advantage of the entire system. This enables incremental upgrade of the system, to include additional metabolic databases but also other biological domains (e.g., genes and proteins databases).

Chapter 3

Standardization of Genome-Scale Models

Abstract

Genome-scale metabolic models are a valuable instrument to study biological organisms. They are capable of performing *in silico* prediction of phenotypes after genetic or environmental changes. These models are built from the functional annotation of the organism's genes that assemble its biochemical network, allowing to expose the enzymatic machinery of the organism.

The Systems Biology Markup Language (SBML) is the most common exchange format to describe these models, and it is widely adopted by many of the existing software platforms within this field.

There are several limitations in the earlier versions of the SBML specification that limits the annotation of many relevant attributes, such as: genes, constraints, reactions and compounds references. Because of that, many tools implemented their own methods to describe these attributes.

In this work, the SBML format is analyzed to create a framework that is capable of

standardizing previous (or alternative) representation methods into the latest standards.

The implemented framework is used to address two case studies to conduct comparative analysis of genome-scale models. The first case study uses the pipeline to standardize 108 models of prokaryotes providing a strong benchmark to test the success rate of the integration methods. The second case study is the unification of *Saccharomyces cerevisiae* models to generate a dedicated database that provides a consensus view of yeast models.

The integration and standardization tools were implemented into a single module in the KBase platform that improves the compatibility of the system with external SBML models.

3.1 Introduction

In the last decades, sequencing technologies suffered many improvements, having the cost of whole genome sequencing decreased significantly, and the availability of fully sequenced genomes increased. This made genome-wide analysis projects more appealing and feasible.

From the genome, it was given the promise to decipher the machinery that keeps cells functioning [95], allowing researchers to exploit the cellular capabilities, but also its limitations, for a broad range of applications, such as, biotechnology, ecology, health, etc.

Functional annotation exploits this genetic blueprint to build computer models to expose these features. The first genome scale metabolic model (GSM) was developed in the year 2000 for *Haemophilus influenzae*[113] allowing a genome wide pathway quantification.

Up to date, genome-scale modeling techniques and tools have been vastly improved. Dedicated software tools provide better integration of many necessary tools to annotate and assemble GSMs [29]. Automated pipelines were developed that are capable to scaffold a GSM by only requiring its genome sequence [99, 6].

The Systems Biology Markup Language (SBML) was created to increase interoperability between existing tools, allowing them to adopt a common exchange format for models.

Although they share the same format, the earlier SBML versions had several limitations regarding annotation options. Distinct research groups adopted their own methods to annotate important attributes in the SBML file, and in some cases in-house extensions were developed.

In a recent work, Chindelevitch et al.[19] claimed that the current state for GSM predictions was inconsistent and an exact arithmetic solver is necessary for reproducibility of the analysis. However, this was later counterclaimed[31], suggesting that most of the problems found for miscalculation of the results was due to incorrectly parsing the models, thus the main issue was the lack of standard representation. In fact, a major drawback of the SBML is the lack of options to specify important attributes related to the annotation of the model components[105], which led to an increasing number of different approaches that were improvised by the modeling community.

For annotation (both metabolites and reactions), there are only a few tools. The Metingear[85] is a Java application that allows automated annotation with support for KEGG, MetaCyc, ChEBI, LipidMAPS and HMDB. The application requires users to setup the databases which might not be practical. Metabolic databases are bulky and in many cases the *loaders* gets deprecated with the update cycles.

Web applications (MetExplore[87], MEMOSys[100]) are more suited for this job since they provide the data infrastructure to support the annotation, however these tools are highly limited to the number of databases supported.

The purpose of this chapter is to explore the representation taxonomy of SBML GSMs and develop strategies for unification and standardization of these models. In this chapter, several interpretation methods were developed to extract relevant GSM attributes from a variety of representation patterns that are found in published SBML models. A standardization pipeline is implemented to unify SBML GSMs to the latest standards in a fully automated approach.

The developed tool was applied to standardize 108 published GSMs of prokaryte microorganisms, providing a large sample of existing representation methods in SBML GSMs and allowing to benchmark the developed tools. The pipeline was able to integrate in average 70% of the metabolites in these models with external database references.

In the second case study, several *Saccharomyces cerevisiae* GSMs were integrated for comparative analysis. As a result, a standalone database was generated to catalog integrated models that represents the metabolism of yeast. The generated database was enriched with additional information regarding yeast genomics.

Finally, an application was developed and made available for public usage in the DOE Systems Biology Knowledgeable (KBase) platform, that allows to standardize SBML models. It includes automated standardization methods (e.g., medium identification, genome strain detection, nomenclature translation), but also manual assignment of other components such as: the biomass reaction, definition of compartments, user defined references of metabolites, reactions and genes.

3.2 Constraint Based Genome-Scale Models

A genome-scale network reconstruction (GENRE) is a compilation of all possible biochemical reactions that are available in an organism's genome. With functional analysis tools, it is possible to extract the metabolic functions of the respective genes to assemble a metabolic network (MN). Therefore, each GENRE is actually a library that compiles all the possible biochemistry of the organisms either inferred from the genome or extracted from literature data.

The detail level of a GENRE is at the reaction level, metabolites are connected to each other by a chain of biochemical transformations. Several efforts are made to increase the detail of the models to add additional constraints, such as gene expression [80], protein structure [12], or regulatory effects [34]. To date, most of the GSMs are solely composed of metabolic constraints since they required less data.

Most of the reconstruction tools attempt to associate gene annotation with the biochemistry of generic pathway databases (mentioned in the previous chapter). There are several approaches to link genes with reactions, such as: extracting the EC number from the genome annotation [28], standardizing the annotation with a controlled vocabulary[24], and protein orthology [1].

However, unlike the previous biochemistry databases, the reactions in genome-scale models must be compartmentalized. The compartmentalization of a reaction defines in which physical space of the cell it is occurring. When the reaction occurs in several compartments it is duplicated in the model, since the enzyme activity occurs in several places. For bacterial models, this problem is simplified since most of these species have only 2-3 compartments (i.e., extracellular space, cytosol, periplasm), but for eukaryotes an additional effort is required to predict the subcellular localization of the enzymes. The transport of molecules between compartments is possible by adding transporter reactions, which require the identification of membrane proteins that are capable to transfer compounds between cellular spaces[27].

The completeness and the accuracy of a GENRE is coupled with the quality of gene annotation in the organism genome. For species with less literature data, most the functional annotation in the genome is most likely inferred by computational methods (e.g., protein similarity).

3.2.1 Constraint-Based Modelling

Constraint-Based Modeling (CBM) is a common alternative to kinetic modeling, which requires mechanistic rates of the equations for each reaction. Kinetic models are hard to simulate and to parameterize, requiring information difficult to acquire to accurately simulate dynamic systems. For large-scale networks, this problem becomes intractable [116]. In CBM approaches, the only critical parameters are the reaction stoichiometric coefficients and reaction bounds, which are much more accessible. The basis of steady state analysis is granted by the assumption that the system is at equilibrium, i.e. there is no change in the concentration of the internal metabolites. This allows to define the feasible space of the system by defining linear constraints, enabling to understand the capabilities of the system under several conditions.

Flux Balance Analysis (FBA) [112, 98] (Definition 12) is one of the earliest CBM methods to explore cellular metabolism. Based on the assumption that the system is in a pseudo steady state, it can be described, by a set of linear equations. In FBA, the organism is assumed to pursue a certain metabolic goal in the form of a linear objective function (3.1), which the FBA method uses to optimize the flux distribution that maximizes this goal. The formulation of the linear program (LP) consists in the stoichiometric matrix (S) of a MN with m metabolites and n reactions. The reaction boundaries are defined by setting a lower and upper bound (3.3) coming from thermodynamics (reversibility), from uptake rates or simply by setting large negative/ positive values (unbounded).

Definition 12. *(Flux Balance Analysis) The FBA is defined by the following LP:*

$$\max \quad Z = \sum_{j=i}^{n} c_j . v_j \tag{3.1}$$

$$s.t. \quad \sum_{j=1}^{n} S_{ij} . v_j = 0 \quad , \quad \forall i = 1, \ldots, m \tag{3.2}$$

$$v_j^{LB} \leq v_j \leq v_j^{UB}, \quad \forall j = 1, \ldots, n \tag{3.3}$$

where Z is the objective function; S_{ij} is the stoichiometric matrix with i metabolites and j reactions; v_j is the flux vector; v_j^{LB}, v_j^{UB} are reversibility constants (reaction lower/ upper bounds).

The most common objective function is the biomass maximization, which represents a proxy for cellular growth. Other functions may also be used, such as maximization of ATP production or of the production of a certain metabolite.

The transformation of a GENRE to a GSM includes the addition of the mathematical components that are necessary to formulate the LP to predict phenotype. Depending on how complete and accurate is the genome annotation, the GENRE may have several reactions missing that leave gaps in the biochemistry. Gap-filling algorithms[106, 55] add missing reactions to connect the GENRE, which is necessary for a functional mathematical model. These algorithms fill the missing reactions using heuristics to predict possible reactions that are needed to fill the gaps (although they have no real evidence).

Having a fully connected network still requires the addition of other components, that are related to modeling purposes.

Biomass Reaction: The biomass reaction is the most common objective function (BOF) in prokaryote GSMs. It is a pseudo-reaction that mimics cellular growth by assembling essential macromolecules (e.g., DNA, RNA, protein, lipids, etc) and demanding their building block compounds (e.g., nucleotides, amino acids, fatty acids, etc) as biomass components [38]. This objective function is based on the assumption that cells evolved to optimize the nutrient consumption towards growth potential. In a flux analysis perspective, the biomass objective tests for the network capability to supply all the components, while still maintaining the steady state constraint. Therefore, the participating components directly influence the complexity of the GSM since a basic BOF demands fewer pathways.

Drain Reactions: The drain reactions allow to unbalance the network allowing model consumption and accumulation of metabolites. These reactions are categorized into three different types of drains: exchange, sink and demand reactions. The separation of the drains is solely used for better manipulation of the model since not all drains have the same meaning.

The exchange reactions are all drains that represent the change of concentration in the media (or environment). These reactions are usually subject to software manipulation to evaluate the model behavior in different media composition.

The demand and sink reactions are auxiliary drains that are necessary for the model to carry a certain flux distribution (e.g., biomass production). As an example, a model may require internal accumulation (in many cases only a residual value) of a certain compound with unknown fate to synthesize a biomass component. It is also a common practice to add sinks and demand reactions for debugging purposes.

Gene Protein Reaction Association: The connection between the genome and the model is held by gene-protein-reaction (GPR) rules. The reactions in the GSM are assigned to genes in the genome that are responsible for the enzymatic activity. Because genes are multi-functional and for certain reactions a multi-enzyme complex is required, a GPR is described as a Boolean expression of the possible gene combinations that are necessary for the reaction to take place.

3.3 Systems Biology Markup Language

The systems biology field is highly interdisciplinary since it bridges several omics technologies to study system wise interactions[59].

Many tools were developed to apply a variety of approaches to model these systems, but most software applications receive as input and produce their own data formats, making difficult to integrate and combine distinct tools. The Systems Biology Markup Language (SBML) [58] was created to address the lack of interoperability between different software tools in the systems biology field.

The SBML is a XML syntax that allows to represent several elements in the systems biology universe. The first version of the SBML (SBML Level 1) was mainly oriented towards dynamic models. Up to date, there are three versions of SBML: Level 1, 2 and 3.

The SBML allows to represent biological entities (*species*) and their interactions (*reactions*) that take place in a container space (*compartment*).

Each *species* represents an entity (e.g., molecule) in a specific state, while *reactions* transform *species* with a defined mathematical rate.

The scope of entities and attributes discussed in this work is oriented towards the representation of genome-scale constraint based models. The main component of a GSM is the stoichiometric network that is defined by the reactions in the model (Definition 12).

The SBML was subject to several updates since the first version (SBML version 1). The main problem of the earlier versions was the limited capability to add annotations to the GSM entities, that are relevant to characterize the molecules and reactions, such as, information regard to genes, mathematical constraints and other important information to aid the users to interpret the logic of the model.

All elements of the SBML inherit the abstract base element `SBase`, that allows other elements in the SBML to inherit a `notes` and an `annotation` sub-element and two attributes *metaid* and *sboTerm*. These attributes were introduced after the SBML version 2 to improve annotation options. The *sboTerm* refers to the Systems Biology Ontology [22] (SBO) being oriented to modeling terms that characterize components of several families of objects found in models (e.g., `SBO:0000243`: gene). It allows categorizing elements in the SBML that play a distinct role in the model. As an example, the biomass reaction is described as a regular reaction in the SBML, while with the SBO annotation these reactions can be flagged with the proper ontology (i.e., SBO:0000629 biomass production), which makes these reactions distinguishable from the remaining of the model.

A metabolite in the SBML is defined as an instance of a `species` (Example 6) identified by the *id* attribute of type `SId`. The `SId` is a string used to identify objects in the SBML, only alphanumeric characters are allowed and it must start with a letter or the underscore character [1]. All other attributes are optional (i.e., *name*, *compartment*, *charge*, *initialConcentration*, etc), but for the later versions of SBML the *compartment* is mandatory since all `species` should be defined in a physical location.

Example 6. *SBML species*

```
<species metaid="EC0027_meta" id="EC0027" name="D-Glucose[e]"
compartment="Extra_organism" boundaryCondition="false" charge="0">
```

A reaction is defined by the `reaction` element, like the `species` the *id* is the only required attribute. The `reaction` element contains four sub elements, the `listOfReactants` and `listOfProducts` that define the stoichiometry of the reaction by referencing the `species`. The `kineticLaw` (Example 7) was the initial method to define the reaction constraints and the objective function. The `listOfModifiers` is less common in a GSM, and when present most of the models use it for a structural definition of the GPR instead of a string annotation. In rare occasions, it also defines an enzymatic catalyst (`SBO:0000460`).

[1]`SId` pattern: `(_|[a-z]|[A-Z])(_|[a-z]|[A-Z]|[0-9])*`

In these cases, the application of ontological terms is crucial for the understanding of the logic of the entities.

Example 7. *SBML Kinetic Law*

```
<kineticLaw>
  <math xmlns="http://www.w3.org/1998/Math/MathML">
    <ci> FLUX_VALUE </ci>
  </math>
  <listOfParameters>
    <parameter id="LOWER_BOUND" value="0" units="mmol_per_gDW_per_hr"/>
    <parameter id="UPPER_BOUND" value="1000" units="mmol_per_gDW_per_hr"/>
    <parameter id="FLUX_VALUE" value="0" units="mmol_per_gDW_per_hr"/>
    <parameter id="OBJECTIVE_COEFFICIENT" value="0" units="mmol_per_gDW_per_hr"/>
  </listOfParameters>
</kineticLaw>
```

The `species` and `reaction` elements are sufficient for the structural representation of the GSM since together they are capable of describing the stoichiometric matrix. This is usually the only requirement to run flux analysis methods. However, interpretation of the model context usually requires additional information. For gene prediction, the reactions have to be assigned with their respective genes (GPRs), while for better interpretation of the metabolic content, the structural information regarding the metabolites should be present because of the ambiguity in compound nomenclature.

The standard SBML supports two methods for extra annotation. The `notes` (Example 8) element allows to declare an HTML body to annotate every element of the SBML, since it is defined in the parent `SBase` abstract class. This would be an easy solution for the annotation problem, and in fact for human interpretation this is sufficient to describe entities in the SBML. However, it is not suited for software tools to parse and interpret the annotation since it is text based, while it is also subject to the proliferation of errors and ambiguous definitions.

Example 8. *SBML Notes*

Annotation of a reaction with `notes` *to specify GPR and Subsystem*

```
<notes>
  <body xmlns="http://www.w3.org/1999/xhtml">
    <p>GENE_ASSOCIATION: (b1377 or b0929 or b2215 or b0241)</p>
    <p>SUBSYSTEM: Transport, Outer Membrane Porin</p>
    <p>EC Number: </p>
  </body>
</notes>
```

An alternative approach is to use the `annotation` (Example 8) element. It provides a structural approach to annotate any SBML element. Usually, the `annotation` mixes the RDF specifications with model qualifiers to provide ontological information. The usage of BioModels.net Qualifiers[78] provides an ontology to specify the meaning of the relationship (e.g., is: identity, encodes: encodement, hasPart: part, etc), which is extremely important since, in most cases, the tools make general assumptions that everything assigned to each other shares a relationship of identity. However, as described in the previous chapter, molecules and reactions may share an hierarchical relationship (since some molecules are sub-instances of others).

The `annotation` element solves the problem of having a structural method to specify the relationship between the object and the annotated instances, but it does not solve the issue of how to provide a standard method to characterize the annotated instance. As an example, in the previous chapter it was described that mapping synonyms of databases is a major bottleneck of the ETL (e.g., KEGG, KEGG Compound, Ligand Compound are all possible synonyms to describe an instance from KEGG). The Minimal Information Requested In the Annotation of biochemical Models (MIRIAM) [75] and later the Identifiers.org [62] provide standard methods to reference external resources. The MIRIAM registry catalogs biological resources providing standard Uniform Resource Identifiers that points to the actual web address of these resources, while Identifiers.org is built on top of MIRIAM to provides an URL for these resources (e.g., http://identifiers.org/ec-code/1.1.1.1 is the URL of the MIRIAM urn:miriam:ec-code:1.1.1.1).

Example 9. *SBML Annotation*

Annotation of a compartment with a GO accession using Identifiers.org with a identity relationship.

```
<annotation>
<rdf:RDF xmlns:rdf="..." xmlns:bqbiol="...">
  <rdf:Description rdf:about="#Extra_organism_meta">
    <bqbiol:is>
      <rdf:Bag>
        <rdf:li rdf:resource="http://identifiers.org/obo.go/GO%3A0005576"/>
      </rdf:Bag>
    </bqbiol:is>
  </rdf:Description>
</rdf:RDF>
</annotation>
```

The Flux Balance Constraints (FBC) is an extension that was introduced in the latest version of SBML (the SBML Level 3) providing structural representation for elements related to constraint based models. The FBC extension includes structural definition for genes and GPRs (Example 10). The genes are defined in a list of entities similar to the `species` and `reaction`, allowing proper annotation of genes in a GSM. The GPR expression was also redefined as an XML elements instead of a string annotation.

Example 10. *SBML FBC3 GPR definition*

Each gene must be defined as an element, this would allow to add additional properties to genes (annotation).

```
<fbc:geneProduct fbc:label="ECSF_3582" metaid="G_ECSF_3582" fbc:id="G_ECSF_3582">
```

```
  <annotation>
    <rdf:RDF xmlns:rdf="..." xmlns:bqbiol="...">
    <rdf:Description rdf:about="#G_ECSF_3582">
      <bqbiol:isEncodedBy>
        <rdf:Bag>
        <rdf:li rdf:resource="http://identifiers.org/ncbigi/GI:281180792" />
        </rdf:Bag>
      </bqbiol:isEncodedBy>
    </rdf:Description>
    </rdf:RDF>
  </annotation>
</fbc:geneProduct>
```

The GPR are represented by XML elements instead of a string and all genes must be declared in the model.

```
<fbc:geneProductAssociation>
  <fbc:or sboTerm="SBO:0000174">
    <fbc:and sboTerm="SBO:0000173">
      <fbc:geneProductRef fbc:geneProduct="G_ECSF_3582" />
      ...
    </fbc:and>
    <fbc:and sboTerm="SBO:0000173">
      <fbc:geneProductRef fbc:geneProduct="G_ECSF_3586" />
      ...
    </fbc:and>
  </fbc:or>
</fbc:geneProductAssociation>
```

Since the previous `kineticLaw` was not fit for constraint-based models, the FBC also defines specialized elements to describe one or more objective functions, while the reaction constraints must also be declared as variables. The lower and upper constraints are defined as the *lowerFluxBound* and *upperFluxBound* attributes in the `reaction` element and they are references to elements in the parameters of the model. A `listOfObjectives` allows to define one or more objective functions, which was a limitation of the previous version.

The FBC is an official SBML extension, but in the mean time, researchers developed external extensions to fit their needs. Most of these extensions were developed to solve problems related to annotation methods that existed prior to the SBML Level 3 and the FBC.

However, there are a few unstructured annotations that were introduced or assumed in the SBML GSM due to the lack of SBML structural support. An important aspect is the explicit definition of the drains (i.e., exchanges, sinks, and demands), since they are highly relevant for the manipulation and interpretation of the model.

3.4 Model Standardization

To accommodate GSM objects in the CDS defined in the previous chapter (Section 2.3.2), the domain is extended with a GSM context. Genome-scale metabolic models belongs to the modeling domain. Like the previous domain of biochemical entities (metabolites and reactions), genome scale models are viewed as a materialization of these abstract objects to actual biological organisms.

Each GSM is represented as an individual *database* in the system, where a unique namespace is assigned to each individual model. The subset of the GSM namespace is defined in $\mathcal{N}_{gsm} \subseteq \mathcal{N}$ that groups all objects in the universal graph database.

Each metabolite species s is an individual object, and like the previous objects it must belong to a unique namespace. In this case all modeling objects must belong to a $n \in \mathcal{N}_{gsm}$, which are the GSM namespaces.

Definition 13. *Model Metabolite Specie / Metabolite / Compartment / Gene*

The metabolite species, metabolites and compartments in GSM are objects $o = \langle id, n, a \rangle$

(Definition 1), such that $n \in \mathcal{N}_{gsm}$.

The set of species $\mathcal{S}$ represents metabolite species of the GSMs, and $\mathcal{M}_{gsm}$ is the set that defines metabolites in models. In GSMs, a metabolite may be defined as several species because of compartmentalization.

The compartments and genes are also materialized as objects and they are defined in the $\mathcal{K}$ and $\mathcal{Y}$ domains.

Definition 14. *Model Reaction*

The reactions in GSM are defined as objects (Definition 1, 13), such that the stoichiometry is defined with a set of edges E that connects the reaction with the metabolite species (Definition 3).

Of all the previous objects the metabolite species and reactions are the only needed to defined the stoichiometric matrix of the model, $\mathcal{G}'_{gsm} = \langle \mathcal{S} \cup \mathcal{R}_{gsm}, \mathcal{E}_{gsm} \rangle$, while the full GSM context is defined in the following subgraph $\mathcal{G}_{gsm} = \langle \mathcal{K} \cup \mathcal{S} \cup \mathcal{M}_{gsm} \cup \mathcal{R}_{gsm} \mathcal{Y} \cup \mathcal{M} \cup \mathcal{R}, \mathcal{E}_{gsm} \rangle$, that contains the domain of metabolite species $\mathcal{S}$, model metabolites, model compartments $\mathcal{K}$, genes $\mathcal{Y}$, model reactions $\mathcal{R}_{gsm}$, metabolites $\mathcal{M}$, reactions $\mathcal{R}$ and model edges.

3.4.1 Metabolites

The standardization of the metabolites requires the identification of the metabolite species in the GSM and the translation of these species to a common nomenclature. The method to standardize the species follows the following steps:

1. Identification of the species context;
2. Expansion of the species annotation;
3. Resolution of internal annotation conflicts;
4. Resolution of overall annotation conflicts;
5. Translation of the species.

The first two are annotation steps, which are related to annotating the species with an identity. The metabolite identity is actually a molecular structure, therefore a proper external database identifier is a common method to describe compounds in GSMs.

The following two steps are related to error correction. Metabolite species that are annotated with multiple identifiers of the same database become ambiguous (assuming these identifiers represent different molecules). This occurs for many reasons, such as problems with the database itself, inaccurate compound structure, integration problems, etc. Cleaning these conflicts is necessary to translate the nomenclatures to a common standard.

Lastly, the standardization of the species involves their renaming to a common naming system, such as the adoption of a single database identification system, or by generating a new unified space.

Each step is detailed next.

Identification of the species context and expanding the species annotation: The first two steps of the standardization process are actually the most important steps of the process. A correct annotation of the species in the model would avoid the need for conflict resolution. The integration of the metabolites in GSM combines two types of functions, ψ and φ.

Definition 15. *Metabolite Species Identification*

The molecular identity of the species is inferred by the $\psi : \mathcal{S} \mapsto \mathcal{M}$. *This function is referred as the species annotation function that annotates metabolite species in the GSM with database identifiers, thus adding chemical detail to the compounds.*

Definition 16. *Metabolite Integration*

The integration function $\varphi : \mathcal{M} \mapsto \mathcal{M}$, *extends the annotation with additional identifiers to cover more database references (i.e., metabolic database integration).*

In a GSM not all species may have a concrete representation, depending on the context some species are mathematical objects for modeling purposes (e.g., the biomass, a protein, a lipid, etc).

The ψ function attempts to guess the species context from the attributes found within the model. In SBML, the species only requires an unique identifier and a declared compartment, the uncertainty level is much higher compared to the integration of database molecules.

Guessing the species context is done by applying several ψ functions, each of these trying to guess based on different approaches (e.g., identifier parsing, name matching, annotation parsing, etc). The Ψ set is a ordered tuple that contains all of the ψ,

$$\Psi = \langle \psi_0, \ldots, \psi_i \rangle. \tag{3.4}$$

The metabolic integration functions φ expand the initial annotation of ψ by using metabolite context, since at this stage information of the annotated references may be inherited by the species. This method is identical to the previous metabolic database integration methods that allows to infer other instances of the same metabolite in different databases.

Unlike ψ, the annotation functions can apply subsequent φ functions, and so the Φ set is a n-ary tuple that contains several ordered sets of φ functions,

$$\Phi = \langle \langle \varphi_0^0, \ldots, \varphi_j^0 \rangle, \ldots, \langle \varphi_0^k, \ldots, \varphi_j^k \rangle \rangle \tag{3.5}$$

where each Φ_k set represents the k-th set of Φ.

The $\zeta^\times$ defines the Cartesian product of Ψ with Φ, this generates the paring between all the functions of ψ and φ (Example 11).

Example 11. *Function Pairing*

$$\Psi = \langle \psi_0, \psi_1 \rangle \;\; \Phi_0 = \langle \varphi_0^0, \varphi_1^0 \rangle \;\; \Phi_1 = \langle \varphi_0^1, \varphi_1^1 \rangle$$
$$\zeta^\times = (\langle \psi_0, \varphi_0^0, \varphi_0^1 \rangle, \langle \psi_0, \varphi_0^0, \varphi_1^1 \rangle, \langle \psi_0, \varphi_1^0, \varphi_0^1 \rangle, \langle \psi_0, \varphi_1^0, \varphi_1^1 \rangle,$$
$$\langle \psi_1, \varphi_0^0, \varphi_0^1 \rangle, \langle \psi_1, \varphi_0^0, \varphi_1^1 \rangle, \langle \psi_1, \varphi_1^0, \varphi_0^1 \rangle, \langle \psi_1, \varphi_1^0, \varphi_1^1 \rangle)$$

Each element of $\zeta^\times$ is actually the composition of the functions ψ and φ that together map species s into subsets of $M \in \mathcal{M}$. The annotation of s is an array of binary values that assigns 1 (or true) for each $m \in \mathcal{M}$ if it is a positive match. The annotation matrix is the concatenation of all annotations arrays given by the combinations in $\zeta^{\times}$.

Definition 17. *Genome-scale Metabolite Integration*

The integration function $\zeta : \mathcal{S} \mapsto \mathbb{Z}^{a \times b}$, is the cartesian product of the Ψ and Φ^k sets, such that, $\zeta = \Psi \times \Phi_0, \times \cdots \times \Phi_k$

For every function in Ψ and Φ_k a confidence value (λ) between 0 and 1 (inclusive) is assigned, and the Ψ^λ and Φ^λ sets map those values to their respective functions (Equation 3.6).

$$\Psi^\lambda = \langle \lambda_0, \ldots, \lambda_i \rangle, \;\; \Phi^\lambda = \langle \langle \lambda_0^0, \ldots, \lambda_j^0 \rangle, \ldots, \langle \lambda_0^k, \ldots, \lambda_j^k \rangle \rangle \in \mathbb{R}, 0 \leq \lambda \leq 1 \tag{3.6}$$

The integration array is the scoring array for every $m \in \mathcal{M}$, where $m_i > 0$ is assumed as positive match.

$$\zeta^\lambda = \Psi^\lambda \times \Phi_0^\lambda, \times \cdots \times \Phi_n^\lambda \tag{3.7}$$

Like $\zeta^\times$, the ζ^λ combines the λ constants, but since $0 \leq \lambda \leq 1$, then the product $\prod \lambda$ must be between 0 and 1. The ζ^λ represents the combined score of the functions in $\zeta^\times$ (since $\zeta^\lambda = \mathbb{R}^{i.k.j}$, where i is the total number of ψ functions and $k.j$ the total number of φ functions).

The matrix $\zeta^\times(s).\zeta^\lambda$

$$\begin{array}{c|cccccc|}
 & \langle \psi_0, \varphi_0^0 \cdots \varphi_j^k \rangle & \cdots & \langle \psi_0, \varphi_j^0 \cdots \varphi_j^k \rangle & \langle \psi_1, \varphi_0^0 \cdots \varphi_j^k \rangle & \cdots & \langle \psi_i, \varphi_j^0 \cdots \varphi_j^k \rangle \\
m_1 & a_{1,1} & \cdots & a_{1,k.j} & a_{1,k.j+1} & \cdots & a_{1,i.k.j} \\
m_2 & a_{2,1} & \cdots & a_{2,k.j} & a_{2,k.j+1} & \cdots & a_{2,i.k.j} \\
m_3 & a_{3,1} & \cdots & a_{3,k.j} & a_{3,k.j+1} & \cdots & a_{3,i.k.j} \\
 & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
m_m & a_{m,1} & \cdots & a_{m,k.j} & a_{m,k.j+1} & \cdots & a_{m,i.k.j}
\end{array} \cdot \begin{bmatrix} \lambda_0 * \lambda_0^0 * \cdots * \lambda_0^k \\ \vdots \\ \lambda_0 * \lambda_j^0 * \cdots * \lambda_j^k \\ \lambda_1 * \lambda_0^0 * \cdots * \lambda_0^k \\ \vdots \\ \lambda_i * \lambda_j^0 * \cdots * \lambda_j^k \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_m \end{bmatrix}$$

defines the annotation scoring for every $m \in \mathcal{M}$ given by the total of $i * k * j$ combinations of the Ψ and Φ functions.

The implemented methods for ψ and φ are described in the implementation section.

Conflict resolution: The integration array of $\zeta^{\times}(s).\zeta^{\lambda}$ may map multiple references of the same database to a single model species, but such occurrences create ambiguity with the identity given to the compounds.

The internal annotation conflicts are all species that are annotated with more than one reference of the same database. As a general rule, for every $s \in \mathcal{S}$ only one metabolite per database can be assigned. These conflicts are solved by selecting the metabolite with higher score.

The external annotation conflicts are metabolites that are mapped to multiple species in the same compartment. It is expected that models have only one replicate per compartment.

Translation: The translation step standardizes the identifiers to a common nomenclature system. A popular practice is to adopt a naming system from one of the metabolic databases since it also allows a closer connection with the database records with the model.

For translation into a database identification system, metabolite species are renamed by the assigned annotation and the compartment identifier is attached as prefix since all identifiers must be unique model wise.

3.4.2 Reactions

The stoichiometry of the reactions in $\mathcal{R}_{gsm}$ is defined by a map function $s : \mathcal{R}_{gsm} \times \mathcal{S} \mapsto \mathbb{R}$. This is a simpler representation compared to the LHS/RHS tuple for the reactions in $\mathcal{R}$ (that are the reactions in metabolic databases) because the stoichiometry of model reactions must have unique metabolite species in both sides of the equation. This is due to the fact that transporter reactions actually operate on two distinct metabolites objects instead of repeating the metabolite.

Reaction annotation The integration of reactions is similar to the method implemented in the previous chapter (Section 2.4.3). However, GSMs contain several transport reactions. In this work, these reactions are excluded from the integration. Transporter reactions are difficult to annotate and catalog since every compound may have a transporter associated, making the catalog of the reactions irrelevant. For the same reason, most of the databases do not contain any transporter reactions with the exception of a few to describe some essential mechanisms of certain pathways.

Merging identical reactions Some models have two different reactions to represent reversible reactions. While the use of this method has some drawbacks it is in some cases inevitable. The disadvantage of having twice the amount of the reactions to define reversibility is the requirement of algorithms to decipher the number of actual reactions in the model, but also makes the model bulky and most softwares to not interpret these reactions as a single reaction but as independent biochemistry.

Cases where it is inevitable to have multiple reactions for the same biochemistry occur when an organism contains several enzymes that are capable to perform the same reaction but some are limited to a certain direction.

Reactions are merged based on their GPR associations and reversibility, the conditions for merging are the following:

1. All reactions have the same biochemistry;
2. All reactions have the same GPR assigned;

   or

1. All reactions have the exact equation.

As an example, if $r_1 = A+B \rightarrow C+D$ and $r_2 = C+D \rightarrow A+B$ have the same GPR (or in some scenarios no GPR assigned at all), it is safe to merge into $r_{1/2} = A + B \leftrightarrow C + D$. If reactions of opposite direction have distinct GPR associations, then this would break the logic of the model, since in the unmerged version it would be possible to knock out the direction of the reaction, while in the merged version this would be impossible.

Combining reactions Many models use pseudoreactions to model the biosynthesis of macromolecules or groups of compounds. In this case, the biomass reaction demands a virtual compound such as a lipid, protein, or in some cases an abstract set of metabolites (e.g., cofactors). These compounds are aggregations of the molecules that are responsible for satisfying their demand.

In a design perspective, this allows better control of the sets, but perhaps also easier manipulation of the model. However, it makes harder for software tools to decipher which are the actual components of these virtual compounds. If these compounds are identified, an equivalent single step reaction can be generated.

For this purpose a set of reaction arithmetics is defined as the following:

- The scalar multiplication of the reaction coefficients $scale(\lambda, r)$ or $\lambda.r$, where $\lambda \in \mathbb{R}$ and $r \in \mathcal{R}$, multiplies the value of each component in the r stoichiometry by λ.
- The arithmetic operation $sum(r_1, r_2, \lambda)$ sums the coefficients of r_1 with the result of $scale(\lambda, r_2)$, this would allow basic subtraction and addition (i.e., $+(r_1, r_2, 1) = sum(r_1, r_2, \lambda)$ and $-(r_1, r_2, -1) = sum(r_1, r_2, \lambda)$).
- The $null(r_1, r_2, m)$ generates an equivalent reaction by merging r_1 and r_2 and removing m from the stoichiometry.

Definition 18. *Reaction combination*

$$null(r_1, r_2, m) = sum(r_1, r_2, \lambda), \text{ where } \lambda = \frac{-1.r_1[m]}{r_2[m]}$$

3.4.3 Drains: Media/Sink/Demand

One of the most common scenarios (typically for GSMs of single cell organisms) is the growth of the cell in a certain environment commonly represented as the growth medium.

The medium defines which of the external metabolites are available in the environment for uptake, in a modeling perspective these are drain reactions as described in the previous section.

A common problem is to identify these reactions in the metabolic network. The drains in the SBML format are represented as any other standard reaction, and in many cases they are difficult to differentiate.

There are several possible methods to define drains, but some representations require manipulation of the stoichiometric matrix before running flux analysis methods. The boundary compounds are usually dead end metabolites defined in a special compartment (the boundary) that encloses the entire network (Equation 3.8). However, for flux analysis methods these compounds must be either removed or drained (adding a further drain to unbalance each boundary compound).

$$A_e \xrightarrow[\text{[-10, 1000]}]{\text{R\_EX\_A}} A_b \tag{3.8}$$

The best drain representation would be a single reaction that contains no products with only the metabolite to be drained (Equation 3.9). The constraints would define if the drain is either for demand, sink or both.

$$A \xrightarrow[\text{[-10, 1000]}]{\text{R\_EX\_A}} \emptyset \tag{3.9}$$

However, in some models the stoichiometry of the drain is reversed, making harder to detect if the metabolite is being accumulated ou consumed (since negative flux in this case is accumulation), but the net flux of the compound would be equivalent.

$$\emptyset \xrightarrow[\text{[-1000, 10]}]{\text{R\_EX\_A}} A \tag{3.10}$$

In reversible scenarios, the drain reactions can also be split into two reactions one for each orientation (Equation 3.11). In these cases the previous merging strategy is applied.

$$A \xrightarrow[\text{[-10, 0]}]{\text{R\_EX\_A\_IN}} \emptyset + A \xrightarrow[\text{[0, 1000]}]{\text{R\_EX\_A\_OUT}} \emptyset \tag{3.11}$$

Finally, some models may couple compounds together for flux forcing (Equation 3.12). In these cases splitting would not generate an equivalent model, therefore these reactions are kept and an individual drain is generated for each of the compounds.

$$A + H \xrightarrow[\text{[-10, 1000]}]{\text{R\_EX\_A}} \emptyset \tag{3.12}$$

The drains are standardized to the single compound representation (Equation 3.9), and renamed to the compound identifier with a drain prefix. The `R_EX_` prefix is assigned for every compound drained in the extracellular space. For other drains, these are renamed to `R_DM_` if lower bound is negative, otherwise it is `R_SK_`. Each of these three drains are also assigned with their respective SBO terms SBO:0000627, SBO:0000628 and SBO:0000632 to represent exchange, demand and sink, respectively.

3.4.4 Gene-Protein-Reactions

In most models, the only connection between the model and the genome is assured by the gene-protein-reaction associations (GPR). The genes in the genome that are responsible for the occurrence of the reaction are assigned by a Boolean expression commonly referred as GPR. This method allows a very simple but efficient way to describe all possible combinations of genes that enable the presence of the reaction (e.g., $gene_1 \vee (gene_2 \wedge gene_3)$).

Before the FBC3 extension, the representation of the GPR was mainly assigned as a string annotation in the notes section. This creates two problems, the difficult interpretation of the notes attribute which was discussed above, and the need of additional software validation for the GPR string.

Equivalent Boolean expressions can be written by switching the order of the terms. To define a standard comparable order for any expression, a normal form is advised be it either the conjunctive or disjunctive. For modeling purposes the disjunctive normal form would be more suitable since it defines all possible gene combinations for the reaction to occur.

To have a standard comparable expression, the GPRs are rewritten in the disjunctive normal form and the variables sorted in lexicographic order.

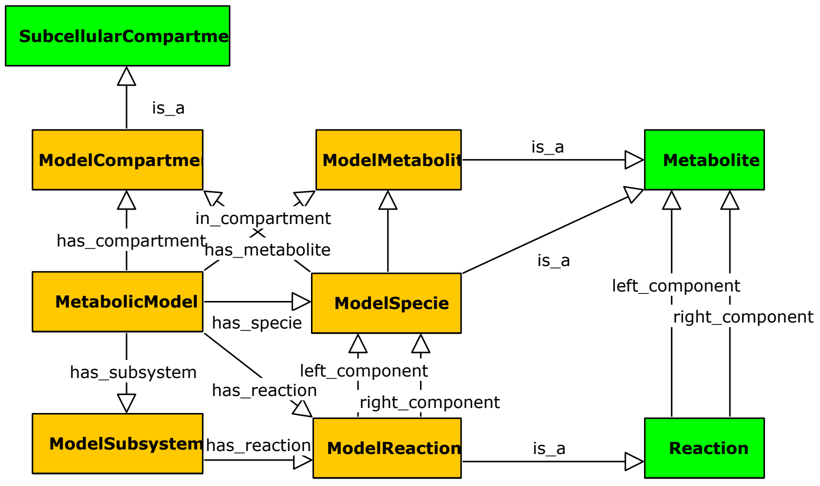


Figure 3.1: Neo4j Labels defined for the CDS namespace domain extended with GSM objects. Metabolite, Reaction and SubcellularCompartment were pre-exiting classes.

3.5 Implementation

The integration of a GSM is similar to the integration pipeline proposed in the previous chapter. The neo4j constraints are extended with the GSM objects (Figure 3.1).

Each GSM is also subject to an ETL pipeline (Figure 3.2), but no cleansing is made to the data, the attributes and annotation (i.e., notes and annotation elements) found within the model are kept as it is. The responsibility to interpret the data is given to the integration method.

For the extraction phase, a SBML reader is implemented from scratch using the existing XML parsing methods. No reaction context is applied to integrate the GSM reactions. The reactions in the GSM may contain unbalanced and custom reactions, which are likely to feed incorrect information, but the reactions in the model are assigned with database reactions and not the reverse. However, models are able to integrated against each other, allowing to infer reactions that are not present in databases but shared between multiple GSM.

The standardization made to the model (includes all automatized fixes and manual changes to the original SBML data) is kept in a separated attribute. This is important to

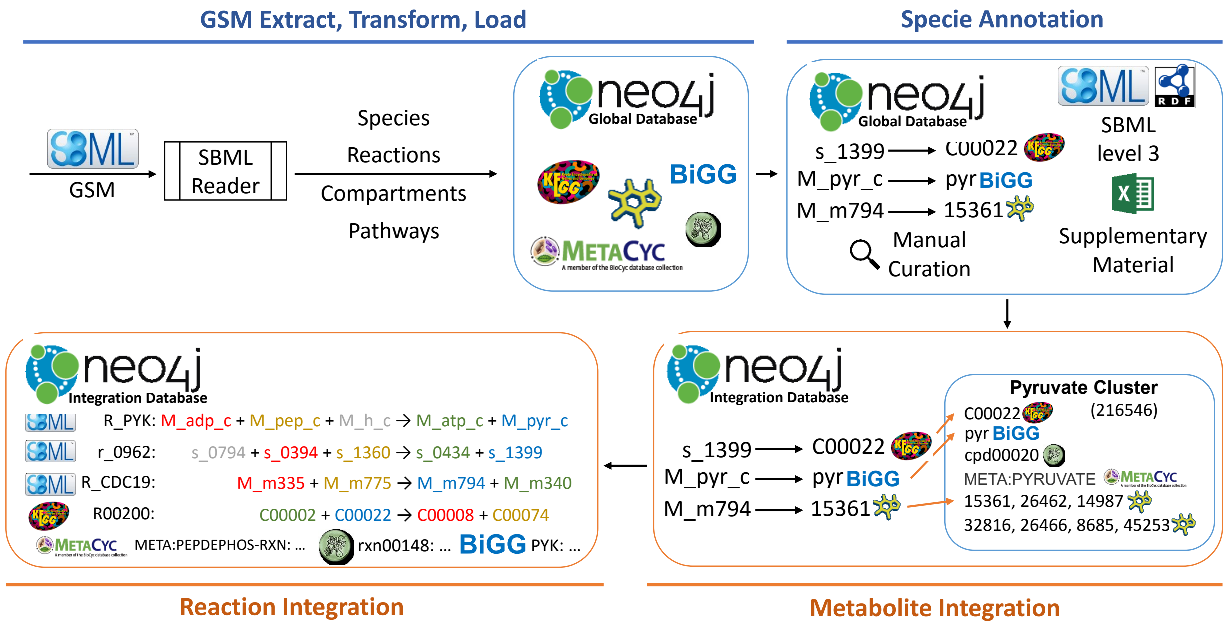


Figure 3.2: GSM integration pipeline. An ETL pipeline loads the SBML into the graph database. The model integration module annotates the loaded objects with the existing metabolic database instances.

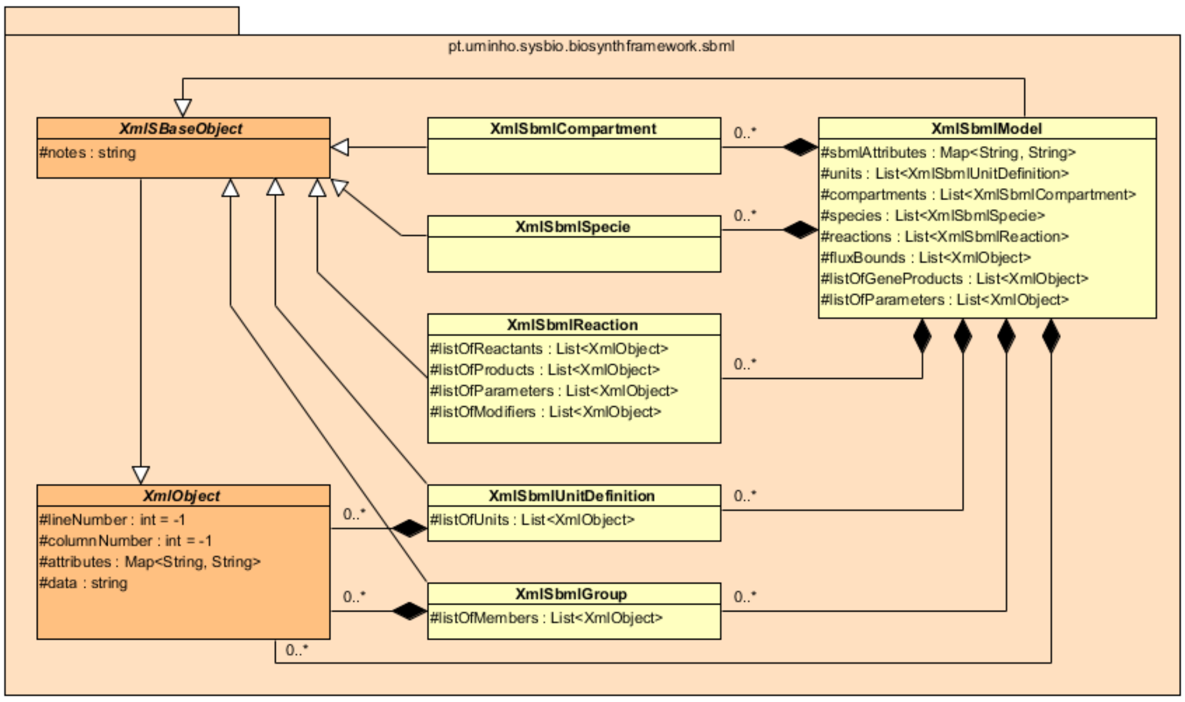

pt.uminho.sysbio.biosynthframework.sbml
XmlSBaseObject
#notes : string
XmlSbmlCompartment
XmlSbmlModel
#sbmlAttributes : Map<String, String>
#units : List<XmlSbmlUnitDefinition>
#compartments : List<XmlSbmlCompartment>
#species : List<XmlSbmlSpecie>
#reactions : List<XmlSbmlReaction>
#fluxBounds : List<XmlObject>
#listOfGeneProducts : List<XmlObject>
#listOfParameters : List<XmlObject>
XmlSbmlSpecie
XmlSbmlReaction
#listOfReactants : List<XmlObject>
#listOfProducts : List<XmlObject>
#listOfParameters : List<XmlObject>
#listOfModifiers : List<XmlObject>
XmlObject
#lineNumber : int = -1
#columnNumber : int = -1
#attributes : Map<String, String>
#data : string
XmlSbmlUnitDefinition
#listOfUnits : List<XmlObject>
XmlSbmlGroup
#listOfMembers : List<XmlObject>
0..*

Figure 3.3: The abstract XmlObject class allows to specify any element of an XML file (the namespace is excluded). The XmlSbmlModel represents the `sbml` element in the SBML file. The reader is implemented for the purpose of parsing all elements that are relevant to a GSM.

preserve the original data for validation purposes, but also to track all changes made to the original model.

3.5.1 SBML Reader, Validation, Profiling

Instead of using the JSBML library [109], a new reader is implemented that parses the SBML as a standard XML. The parser is implemented in Java using the *XMLEventReader* to parse all XML elements. This will allow to read any SBML file even if it violates any of the SBML schema definitions as long as the XML is properly structured.

Since the SBML provides support for a wide range of systems biology models, the reader was implemented to only read components related to constrain-based modeling (i.e., stoichiometric models for flux analysis).

The base object is an abstract *XmlObject* (Figure 3.3) that stores the line and column of the XML element, its attributes and the body. In SBML, the elements usually do not contain data.

For every element, a specific interpreter is implemented, while elements without an interpreter are skipped and a report is provided to notify all unprocessed XML objects.

The data structure for the read components from the SBML follows a hybrid generic XML format with some SBML syntax. Every object in the *XmlSbmlModel* inherits the *XmlObject* that contains information about the line and column, the attributes and the data of the element.

Compartments, Species, Reactions The basis of a GSM is the metabolic network of reactions. The definition of the three core components is specified in three lists in the SBML (`listOfCompartments`, `listOfSpecies`, `listOfReactions`). An element parser is dedicated to each of the list objects (compartment, species, reaction).

External references detection: The `notes` and `annotation` are the elements that are searched for external references. To parse references from the `notes`, a dedicated parser is implemented to transform the HTML information into a map $notes : k \mapsto v$, that for every representation of key k in the HTML body, maps to its corresponding string value v. As default, the ”`:`” symbol is used as split separator for the *key* and *value*, and it is assumed that every record is written in a single HTML paragraph (i.e., between the paragraph elements `<p></p>`). An additional mapping function is needed to decipher the meaning of each *key*, to map these into the database namespaces (declared in the universal database $\mathcal{G}$ previously).

Unmapped elements are given a warning to notify unusual keywords.

The `annotation` element is easier to parse since the XML element iterator is enough to pass through all elements. The reader extracts only elements enclosed by the `is` BioModels Qualifier since it is looking for the identity of the compounds and reactions. Other qualifiers may be implemented in the future, but there is little use for additional ontology at the current state of the methods.

GPR detection: The GPRs are extracted from three distinct places: `notes`, `modifiers`, and `geneProductAssociation`. The method to extract from the `notes` is identical to the one used for references defined above. The HTML block is converted into a simple map and a mapping function is required to decipher the *key* that corresponds to the GPR expression (the default keyword is *GENE_ASSOCIATION*). If the *value* is present it is subject to an expression parser to validate but also extract the genes. The parser is taken from the OptFlux metabolic engineering software [107].

The second approach is to detect the GPR expression from the modifiers. Many tools adopted the `listOfModifiers` element which is a child of *reaction* to provide a structured method to represent a GPR. In these scenarios, each `modifierSpeciesReference` element references a `species` element that represents a gene while together they assemble the *or* expression. To define the *and* expression a *species* is declared to represent the enzyme complex and a `reaction` must be declared to consume the required genes in order to produce the enzyme complex. The reader reverse engineer these occurrences by generating the equivalent GPR expression and flags these reactions and compounds as genes since they influence the total numbers with a naive reader.

The last approach is to extract the FBC `geneProductAssociation`. The assembly of the GPR expression from the FBC is trivially achieved by iterating the XML elements. The only validation required is to verify if all genes are declared in the SBML.

3.5.2 Metabolite Integration Methods

In this section several implementations of ψ and φ methods are described. The following methods define implementations of ψ:

Metabolite identifier: The compound identifier is a mandatory attribute for all compounds in the model. In many cases, it is common that the identifier is inherited from a metabolite identifier of a metabolic database (commonly the one used for model reconstruction).

Since the identifier must be unique for each species, to represent the metabolite in different subcelullar locations, usually a suffix is added. By a general rule, this suffix is usually the identifier of the compartment. Also, the prefix `M_` is commonly given to every compound since the SBML specification disallows `species` elements to start with a symbol or a number.

Let $\mathcal{A}_s$ be a set of all identifiers of metabolic compounds in a model, then if $s_s \in \mathcal{A}_s$ represents an external database identifier $i \in \{id(m) | m \in \mathcal{M}\}$, then i is a substring of s. However, there may exist several database identifiers that are substrings of s since it is possible that database identifiers be substrings of each other (e.g., *o2* is substring of *h2o2* in the BiGG identifiers). In this case, the largest i that is a substring of s is accepted, in scenarios where two distinct identifiers of same size are the highest match they are both discarded.

The Aho–Corasick algorithm[2] is used for fast substring matching. The algorithm assembles a finite state machine that is a prefix tree from a finite set of strings (or dictionary). The integration method builds a prefix tree from a set of identifiers (usually from a single database to avoid conflicts).

For each $s \in \mathcal{A}_s$ that matches i in the identifiers set, the prefix and suffix of the unmatched portion are extracted (e.g., given $s = M\_o2\_c$ and $i = o2$, the match generates a prefix $M\_$ and suffix $\_c$). After matching every element in $\mathcal{A}_s$, the prefixes and suffixes frequencies are calculated and, for every match that generated a prefix or suffix with frequency lower than a reject threshold, it is assumed as a false positives.

Name Dictionary: In many cases, the model inherits the names of the database metabolites. Name matching receives a dictionary with compound names assigned to database identifiers, and only exact string match is accepted.

Model References: The reference function annotates species by extracting the existing annotation found within the SBML. The fields scanned for references are the **notes** and **annotation** elements as described above. The method requires a translation map $t : k \mapsto \mathcal{N}$ that maps the keywords k to database namespaces of metabolites in $\mathcal{M}$.

Manual Annotation: The manual annotation function receives an external file and assigns their respective metabolite species. The purpose of this function is to provide manual curation but also allows to receive feedback from external data such as annotation from the supplementary files of publications. It applies a direct mapping function $t : s \mapsto \mathcal{M}$ that maps species identifiers s to metabolites in the database.

In this work, only a single φ function was implemented:

Reference Expansion: The references are expanded using the integrated metabolite space provided by $\mathcal{I}$ an argument. The $\mathcal{I}$ (Definition 8) is a set of metabolite sets, which represent the clustering of identical metabolites from multiple sources (i.e., databases). This integrated metabolite space can be either generated using the database integration methods in the previous chapter, using external sources (e.g., other integrated databases such as MNXRef), or by using a manually curated set.

The reference expansion function is highly flexible since it can receive any cluster set of integrated metabolites $\mathcal{I}$, and it is does not have to be a fully integrated set since it will only expand references with the information given by $\mathcal{I}$.

This allows to combine the confidence scores λ with the layers of Ψ to tune the integration, by using different sources of $\mathcal{I}$ to generate a consensus integration.

3.5.3 KBase Application

The work in this chapter is implemented in the DOE Systems Biology Knowledgebase (KBase) platform[2]. KBase is an open source data platform that provides computing infrastructure to perform large-scale genome functional analysis.

[2]`http://kbase.us`

The workflow in KBase is done by manipulating a narrative interface that is implemented over the jupyter notebook framework. Users can assemble their pipeline by combining existing applications that are available in the application catalog.

The SBMLTools[3] module was implemented, providing two applications to improve the compatibility of the existing ones with external SBML models. The first application is dedicated to import SBML models, while the second provides a more customized interface to integrate and modify model components, which were not possible to automatize (e.g., compartments, malformed GPR, etc). This allows users to configure and adapt the imported models even if they do not fit the needs of the KBase system. At the time of writing, the module is available in the beta catalog.

KBase Development Environment The KBase SDK (kb-sdk) provides basic tools to configure and scaffold the initial configurations of a KBase module. The modules are KBase application projects that may implement several functions, that are run in a Docker container. A module development life cycle consists of three stages: development, beta, release. The initial state (development) is only exposed in the appdev environment[4] of the platform. This is an isolated environment for the purpose of testing, and it provides access to application publishing options.

Once the module is functionally ready, it can be migrated to the beta stage, that becomes available in the general platform. Finally, to release the module, a request must be sent to the KBase system for release approval.

The SDK tool provides helper functions to configure the application, the `kb-sdk init` function allows to scaffold the initial module layout and its base configuration (Figure 3.4). The platform supports several technologies (python, Java, R or Perl) to build the modules, although each module can work only with one of these. The SBMLTools module is implemented with the Java configurations.

The spec file in the root directory allows to specify the data types, the functions with

[3]`https://narrative.kbase.us/#catalog/modules/SBMLTools`

[4]`http://appdev.kbase.us`

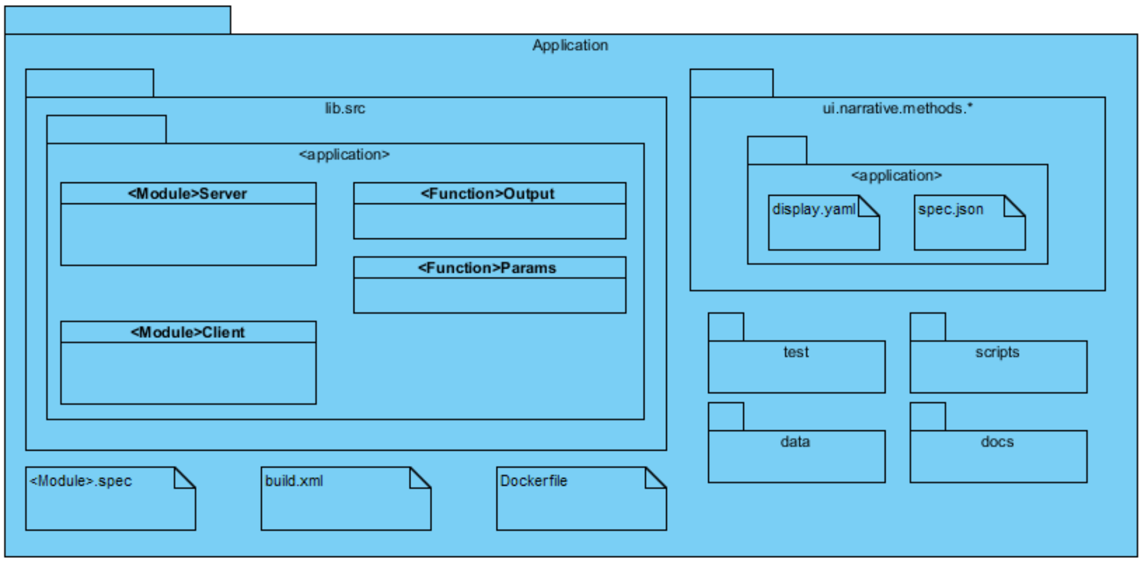


Figure 3.4: The layout of a KBase module. The default configurations are automatically generated with the SDK program.

the input and output data structures available in the module, as for Java projects the specified data types are generated with the jsonschema2pojo to generate the object class files. The spec file uses a domain specific language implemented for KBase that is similar to the type definitions in the C language and the JSON schema. The module contains a single Server and Client file that provides an interface between the platform engine and the program logic. Every function defined in the spec file generates a function in the Server and Client file. The function defined in the Server file holds the logic of the program, while the Client functions provide the remote procedure call functions for the client (the platform itself or other modules).

A narrative application interface is declared in the narrative methods sub-folder for each function to be exposed. Not every function defined in the module must be a graphical application in the application catalog, but to turn these functions into applications a narrative specification must be declared. Each narrative application is specified by two files, the display.yaml declares the HTML text of the interface while the spec.json maps the input and output parameters of the server with the interface widgets.

The interface configuration is limited to the widgets implemented in the KBase narrative system. Currently, it supports text fields, text blocks, check boxes, combo boxes and combined groups.

The input and output arguments of the widgets can be integrated with the narrative workspace. As an example, a text field can be filtered to select only objects of a certain type (e.g., FBAModels) in the workspace for easier usability.

Model Import The "Import model SBML from web" application allows users to import SBML models into the workspace (Figure 3.5). This application transforms SBML files into KBase genome-scale model objects (`KBaseFBA.FBAModel`) and saves them in the users workspace. The application also provides automatic annotation of the SBML compounds and reactions.

Because of the limitations of the narrative interface specification, to import a model the application must fetch the model from a HTTP web link (e.g., Github, personal web storage, etc) since no file upload widget is allowed.

The only required field for the importer application is the SBML URL, all other options are optional. The BOFs can also be assigned in this application, but this is optional since it can be done in the follow up application.

By default, the importer application performs automatic annotation of the compounds and reactions.

KBase Nomenclature Integration The "Integrate Imported Model into KBase Namespace" application is the follow up step to integrate external GSMs into the KBase system. This application allows to perform many tasks to reshape external models to fit the KBase platform.

In the KBase system, some compartments have logical meaning, and the most relevant compartments are the extracellular space and the cytosol. The extracellular space is required to attach the media for flux analysis methods. The application does not provide any automated guessing of the compartments in the model. It its up to the user to specify

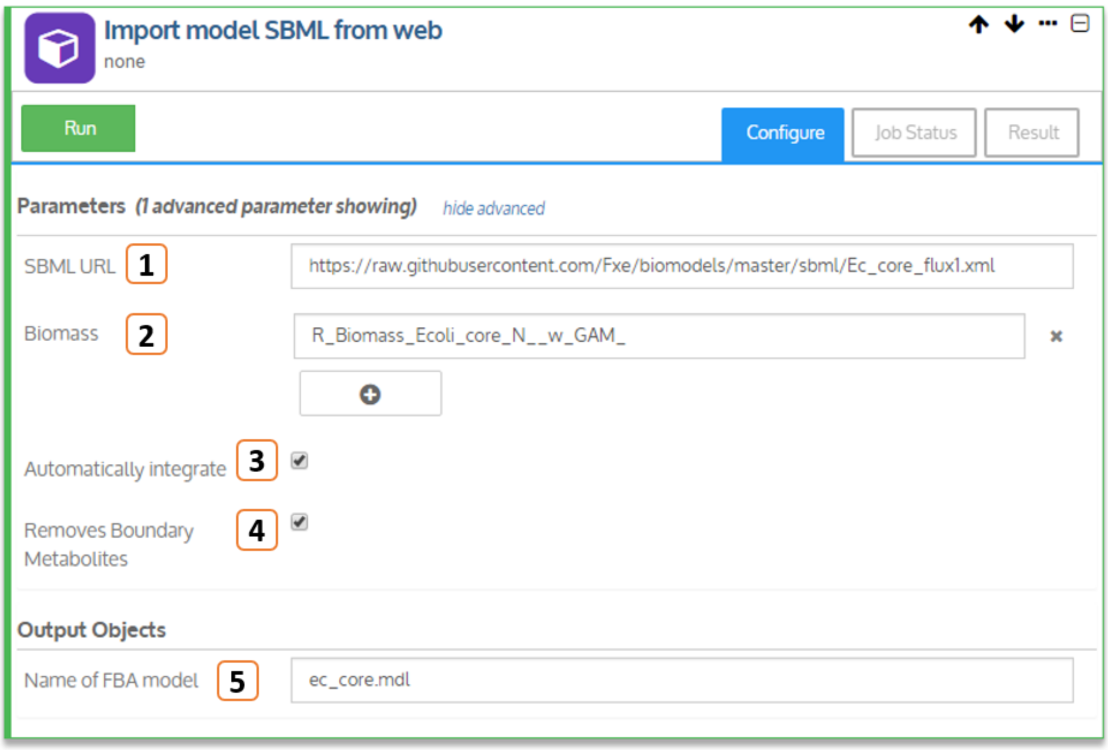

Import model SBML from web
none
Run
Configure
Job Status
Result
Parameters (1 advanced parameter showing) hide advanced
SBML URL 1 https://raw.githubusercontent.com/Fxe/biomodels/master/sbml/Ec_core_flux1.xml
Biomass 2 R_Biomass_Ecoli_core_N__w_GAM_
Automatically integrate 3 ✓
Removes Boundary Metabolites 4 ✓
Output Objects
Name of FBA model 5 ec_core.mdl

Figure 3.5: Interface to import SBML files from the web in the KBase application. 1 - Web URL to download the model (XML or ZIP for multiple import). 2 - Specification of the biomass reaction. 3 - Perform annotation when importing (both compounds and reactions). 4 - Delete compounds with boundary condition assign as *true*. 5 - Output KBase object name.

the GSM compartments using the interface options (Figure 3.6) to assign the imported compartments to the KBase standards.

In the KBase system, the media and the GSM are separated into two different objects. This allows users to configure and share the media independently from the models.

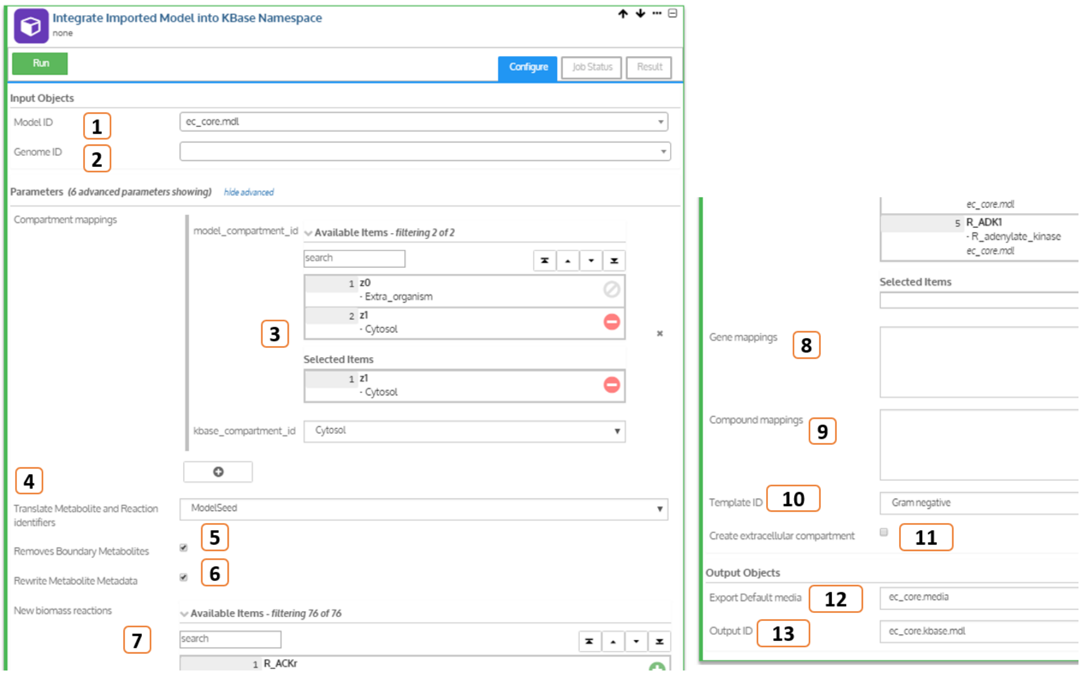
The application allows the user to specify the medium object in the output field. This is not a mandatory field, however if specified, it will strip all the drains in the model and generate a KBase medium object to the workspace. If the extracellular compartment is integrated, only these drains are stripped to the medium, and the others are assigned as demand and sink reactions. The object also inherits the default exchange constraints specified in the model.

The imported GSM is assigned by default to an empty genome. All GSM models in KBase are attached to a genome object, this integrates the genes of the model with data from the genome features.

The application allows to assign a genome object from the workspace to be integrated with the model. It will attempt to match the features in the genome with the genes in the GPRs of the model. All genes in the model that were not possible to integrate with the assigned genome are removed and reported in the output.

The KBase platform indexes the entire set of prokaryotic genomes of the NCBI Reference Sequence (RefSeq) database[104]. The Apache Solr search server is used to create a search index for many genomic features, that includes annotation, sequences, aliases and taxonomy. The *search_kbase_solr* function of the *KBSolrUtil* module is used for automated detection of the GSM genome, this option is enabled if no genome is manually specified. The application requests a Solr query to retrieve all gene features with identifiers or aliases that match each gene in the GPR expressions from the model. The genome that covers most of the genes is automatically assigned to the model, and a copy is given as output to the users narrative. Similar to manual genome assignment, the genes that do not match any feature in the assigned genome are removed from the model and reported. In this case, other genomes (if any) that also had matches, are reported with a matching percentage.

Manual configuration of the model features is also possible in the advanced options. It



Integrate Imported Model into KBase Namespace
none

Input Objects
Model ID: ec_core.mdl
Genome ID:

Parameters (6 advanced parameters showing)
Compartment mappings
model_compartment_id
Available Items - filtering 2 of 2
search
1 z0 - Extra_organism
2 z1 - Cytosol
Selected Items
1 z1 - Cytosol
kbase_compartment_id: Cytosol
Translate Metabolite and Reaction identifiers: ModelSeed
Removes Boundary Metabolites: ✓
Rewrite Metabolite Metadata: ✓
New biomass reactions
Available Items - filtering 76 of 76
search
1 R_ACKr
ec_core.mdl
5 R_ADK1 - R_adenylate_kinase ec_core.mdl
Selected Items
Gene mappings:
Compound mappings:
Template ID: Gram negative
Create extracellular compartment: ☐

Output Objects
Export Default media: ec_core.media
Output ID: ec_core.kbase.mdl

Figure 3.6: Interface to integrate external models into the KBase namespace. 1 - KBase FBAModel to integrate (from workspace). 2 - KBase Genome to integrate with the model (empty for automated detection). 3 - Manual assignment of the compartment logic. 4 - Compound and reaction renaming (ModelSEED / KEGG / BiGG). 5 - Delete boundary compounds. 6 - Rewrite compound names / formula / structure according to the ModelSEED annotation. 7 - Biomass reaction selection (from the selected FBAModel in option 1). 8 - Manual GPR rewrite. 9 - Manual ModelSEED annotation. 10 - KBase model reconstruction template (gram negative, gram positive, core and plant). 11 - Generate KBase Media object from drains (empty to keep drains). 13 - Output integrated FBAModel name.

includes manual assignment of the GPR with the reactions and the manual annotation of the compounds.

The translation option allows to rename the compounds and reactions with either ModelSEED, KEGG or BiGG identifiers from the annotation.

Like the importer application the BOF can also be reassigned, but in this stage model context is available, that allows users to select and search a reaction from a list of reactions that are found in the model.

3.6 Case Study: Prokaryotes GSM integration

Over the last years, hundreds of bacterial GSMs were manually curated and some had several rounds of reannotation and reconstruction. In this case study, over 100 prokaryote GSMs were selected for a semi-automated integration, where the main purpose is to assess the BOF of these models for the presence or absence of essential components. The models were collected by Xavier et al. [127] for a follow up research in the biomass formulation of prokaryote GSMs.

In a prokaryote GSM, the biomass reaction is usually the primary and only objective function which allows to determine if the cell viability in different conditions after environmental and genetic modifications.

There is a high discrepancy between GSM BOF, and many models have several BOF for debugging, different conditions, etc. In a genome-scale scope most of the BOF contains nucleotides, amino acids, and fatty acids for the DNA, RNA, protein and lipids but regarding to ions, organic cofactors and prosthetic groups these are usually missing.

The enzymes that are related to biosynthesis or transport of these compounds usually have less literature content, which also implies that the annotation and the knowledge regarding the origins of these compounds are less to be known, making many GSM miss reactions regarding these compounds, and in some cases the entire pathway is not present.

To assess the models, first a SBML profiling is conducted to detect possible inconsistencies in the SBML structure, followed by the standardization pipeline to unify the

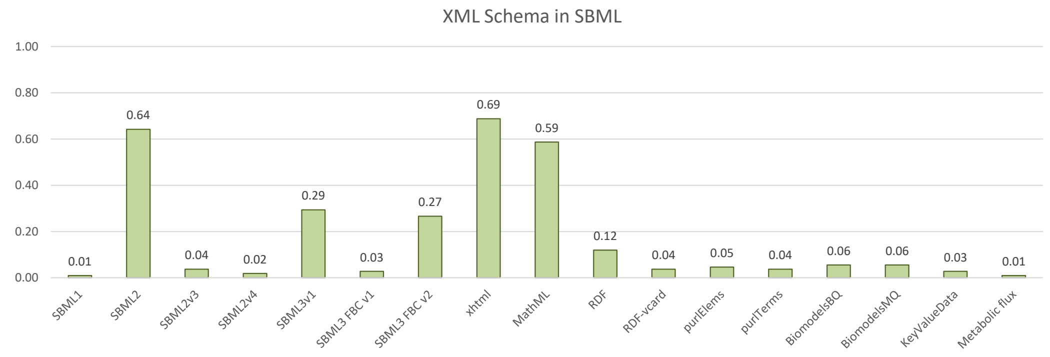


Figure 3.7: XML Schema's included in the 108 imported models. Annotation methods using `notes` and `annotation` element corresponds to the usage of the xhtml and RDF XSD, respectively. The KeyValueData and Metabolic Flux XSD are unofficial SBML extensions.

metabolite and reactions nomenclature.

3.6.1 Model Characterization

The models are interpreted with the implemented reader and the COBRApy reader for comparison purposes. From the 108 models, 13 SBML files were not accepted by the COBRApy library for several reasons: missing identifier attributes, non-unique identifiers, problems building the stoichiometric model. The implemented reader was able to read all the models since the only requirement is a syntactically valid XML file, but depending on the severity of the detected issues some models must be subject to user curation in order to be viable. The reader is able to fix some issues presented with the COBRApy reader such as duplicate or missing reaction identifiers, since for reactions these can be generated automatically. User intervention was required to fix reactions that lacked the `species` attribute in the stoichiometry.

An initial SBML profiling shows a variety of XML XSD included in the models (Figure 3.7). The most popular SBML version is the SBML Level 2, which is perhaps related with the time frame when these models were developed, but also with the use of model reconstruction tools.

The versions before SBML3 did not have any semantics to represent genes and flux constraints. Most of these models relied on ad hoc representation of these parameters using the generic field to write HTML notes, and the kinetic parameters fields. The problem with such methodologies is the need to define custom keywords to describe the properties, and nothing forbids different tools from using different notations to describe these properties, hence losing the whole purpose of interoperability between software platforms/tools.

A few unofficial community developed SBML extensions were also found (Figure 3.7). The Metabolic flux model annotations [115] enable reactions to annotate flux analysis results (such as FBA solutions or other methods), but also include the constraints used, which improves the reproducibility of the analysis. The KeyValueData is another community extension to add a dictionary data structure for the `annotation` element, avoiding the usage of the `notes` element to add arbitrary attributes. It is supported in the PySCeS [96] framework for cellular modeling.

For a correct interpretation of these models, the parsing of these extensions was implemented in the reader even though their frequencies are low.

On average, 70% of the metabolites were able to be integrated (Figure 3.8), however some models showed very low integration ratio, in many cases these are models without any compound annotation while having numerical generated identifiers.

Models that include genes as compounds compromise the total number of metabolites but also the annotation. The iSB1139 GSM shows a 65% of integration, having more than 50% of species annotated with BiGG. In fact, the total numbers are much higher since 30% of the metabolites are representing genes. This was observed for only three models: iJC568, iMLTC806cdf and iSB1139, each of these models would have annotation above 90% if genes compounds were removed from the model.

Models that use custom identifiers without providing proper names and annotation often include additional information in the supplementary materials. However, to properly annotate these models, such information must be transferred to the SBML and in many cases the supplied data cannot be directly transferred since they are not organized in regular tables (the addition of intermediate headers and comments implies dedicated parsers). The

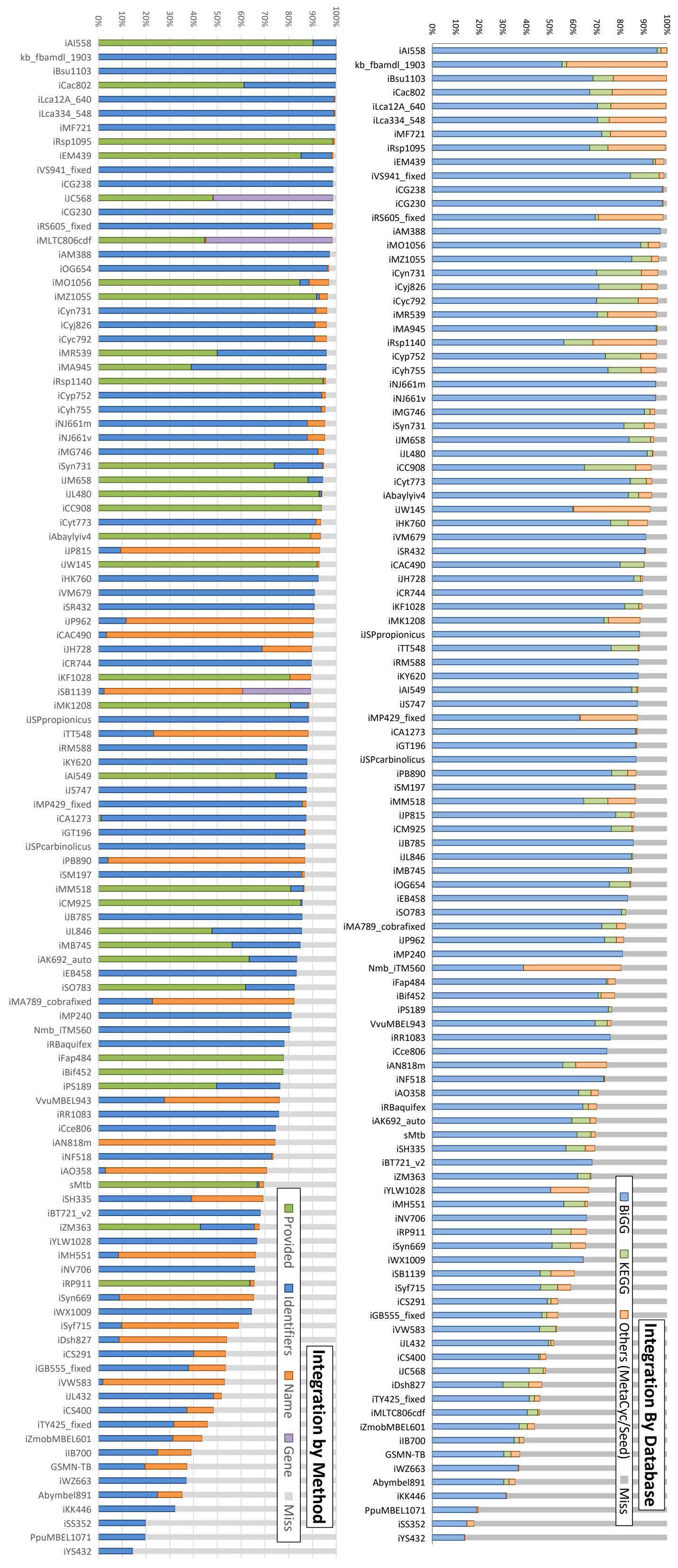


Figure 3.8: Annotation result of the 108 models. The database coverage is ordered by database priority: BiGG, KEGG, MetaCyc/ModelSEED. The method is ordered by source priority: provided, identifiers and compound names

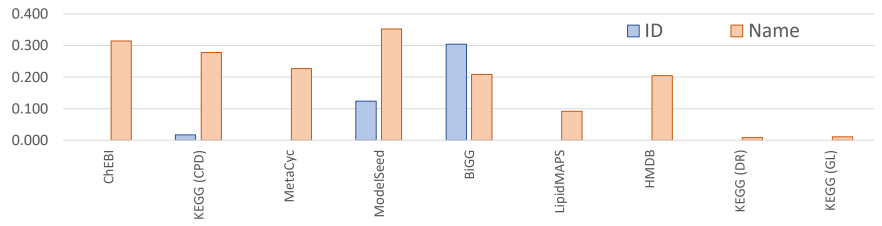


Figure 3.9: Contribution of each database to identify *ids* and *names* of the model *species*.

iSS352, iKK446 are examples of these models, since they are not provided in SBML but assembled from spreadsheets.

In general, when the identifier annotation is low, it implies that these models do not follow database identifiers, but instead they use their own abbreviations. The few matches are all occurrences of commonly shared identifiers that usually match with the BiGG naming system (e.g., adenosine triphosphate is commonly abbreviated as ATP).

The iYS432 identifiers did, however, found to be originated from the MetaCyc database, but minor modifications were made to the identifiers because of the dash "-" character, that was replaced by the underscore. This provides room for improving the identifiers method to include MetaCyc identifier search without dashes. Nevertheless, the MetaCyc identifiers shown to be the least popular for model reconstruction.

The identifiers method was able to detect in average 30% of the metabolite species across all models (Figure 3.9), while the BiGG database was the most adopted nomenclature followed by ModelSEED, and together they cover 40% of all identifiers in 108 models. For readability sake, the identifiers from BiGG are usually preferred since they are abbreviations of compound names. Regarding to names, the ModelSEED is the most popular followed by ChEBI, KEGG (compound) and MetaCyc.

3.6.2 Flux Analysis

Correct interpretation of the mathematical attributes of the model is essential to obtain the correct results from the flux analysis methods. In some models, default assumptions may break the model. In most scenarios a zero growth is easier to fix than having an

Table 3.1: Manual corrections applied to GSM.

Model	Action	Scope	Action
iHK760	Remove _b	species	delete
iJW145	Remove _b	species	delete
iCA1273	Remove _b	species	delete
iRsp1095	Free Exchange	exchanges	Bound $\rightarrow]-\infty, \infty[$
iRsp1140	Free Exchange	exchanges	Bound $\rightarrow]-\infty, \infty[$
iGT196	Keep Boundary	species	keep
iLca12A_640	Free Sink	exchanges	Lower Bound $\rightarrow -\infty$
iLca334_548	Free Sink	exchanges	Lower Bound $\rightarrow -\infty$
iLca12A_640	Manual Uptake	exchanges	Upper Bound
iLca334_548	Manual Uptake	exchanges	Upper Bound

incorrect numerical value since this would require additional knowledge about the model to understand the proper flux ranges.

Only a few models had SBO terms annotated, while for most models that did contain ontology terms these were annotated in the `parameter` element[5]. In the `species` element (compounds), three models had the enzyme, protein complex and metabolite terms[6] to distinguish the elements between compounds and genes. As for reactions, no terms were found to help identify the biomass or drains.

The biomass reaction was manually selected by Joana Xavier. Given these reactions it is essential that their components are standardized into a single nomenclature system for comparison purposes. The integration pipeline was ran to translate all the metabolites into the BiGG universal compound nomenclature for ease of interpretation. It was also necessary to standardize the compartment naming system, since these prokaryote GSMs often include only 3 or 2 compartments; with the exception of the cyanobacteria, all the compartments were manually curated.

Manual modifications were made (Table 3.1) in a few models, including manual as-

[5]SBO:0000625: flux bound and SBO:0000626: default flux bound.

[6]SBO:0000014:enzyme, SBO:0000297:protein complex and SBO:0000299: metabolite.

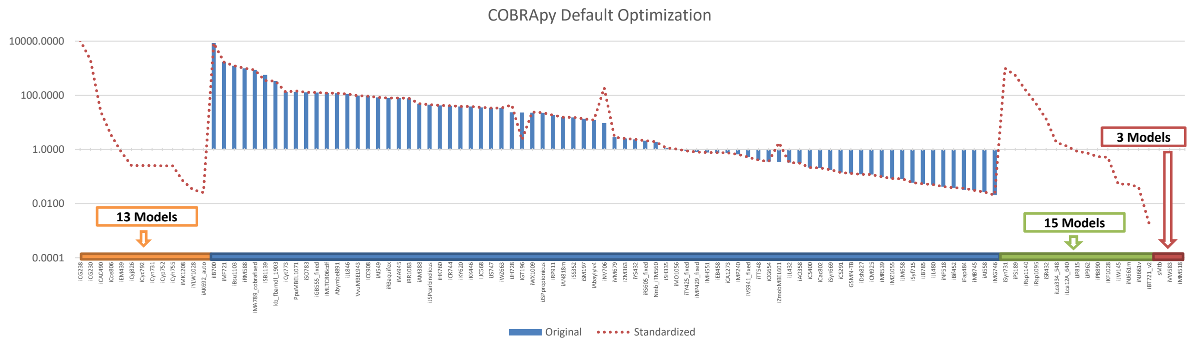


Figure 3.10: Default read and optimize test using the original and standardized versions.

signment of the exchange fluxes, removal of metabolite species ending with a prefix, and allowing boundary species.

To test the reader interpretation and standardization, a flux balance analysis was ran for each model using the default media and the selected biomass reaction. The COBRApy library is used with both versions of the model (the original and standardized version), the only modification made was the selection of the biomass. The *cobra.io.read_sbml_model* function call is used without any parameters to read the model. It is understandable that, for some models the specification for removal of certain compounds is necessary (usually ending with the _b suffix), but the purpose of this test is to assess the default interpretation of the models. To run flux analysis the *optimize()* function is used and the *objective_value* attribute is used to compare the results.

For most of the models that were able to optimize using both versions, the solution remain equal for both with only a few exceptions. The standardized models were able to enable default optimization for 15 models that gave zero growth flux using the original version. Additionally, 13 became parseable after fixes (some required manual intervention) and were able to obtain default growth rate for the selected biomass. Only 3 models remained with zero growth for both of the versions.

3.6.3 Genome Integration

The integration of the GSM with its genome is often complicated due to several facts:

- In some models, the GPR associations are not transfered to the SBML (in these cases they are usually described in publication supplementary data). A possible reason is the limitation of the tools used that are not capable of writing the GPR expressions in the SBML.
- The genome may suffer updates from the time of the publication. These changes can be the entire merge of an organism with another (e.g., duplicated species), changes in the gene nomenclature or removal of genes.
- The ”standard” identifier of the genes may be subject to change.

With the integration application implemented it was possible to automatically detect the genome for 61 models (Figure 3.11), that is a total of 51 distinct genomes (since some have several models). The total amount of metabolic genes vary between organism and the total number of open reading frames in the genome.

In a previous study[71], the authors show that for prokaryotic species the number of metabolic genes may range on average between 15% to 20% of the total amount of genes.

The numbers found in the 61 GSM correlate well with these results. Some models shown significant improvements from earlier versions, the *Clostridium acetobutylicum ATCC 824* (iJL432, iCAC490 and iCac802) display an increase from 10.9% to 18.6% of metabolic genes and *Streptomyces coelicolor A3(2)* (IB700 and iMK1208) with an increase from 8.5% to 15.5%.

Most models had a few genes that were not found in the genome, in many cases fictitious genes. The *s0001* was present in many models to represent spontaneous reactions, another example is a fictitious gene to represent gap filled reactions, but these scenarios are dependent on the application used to build the model. Some models may include genes from another close organism, where it is likely that the model was built using comparative genomics. Nonetheless, the number of genes not found are residual compared to the total genes in the model.

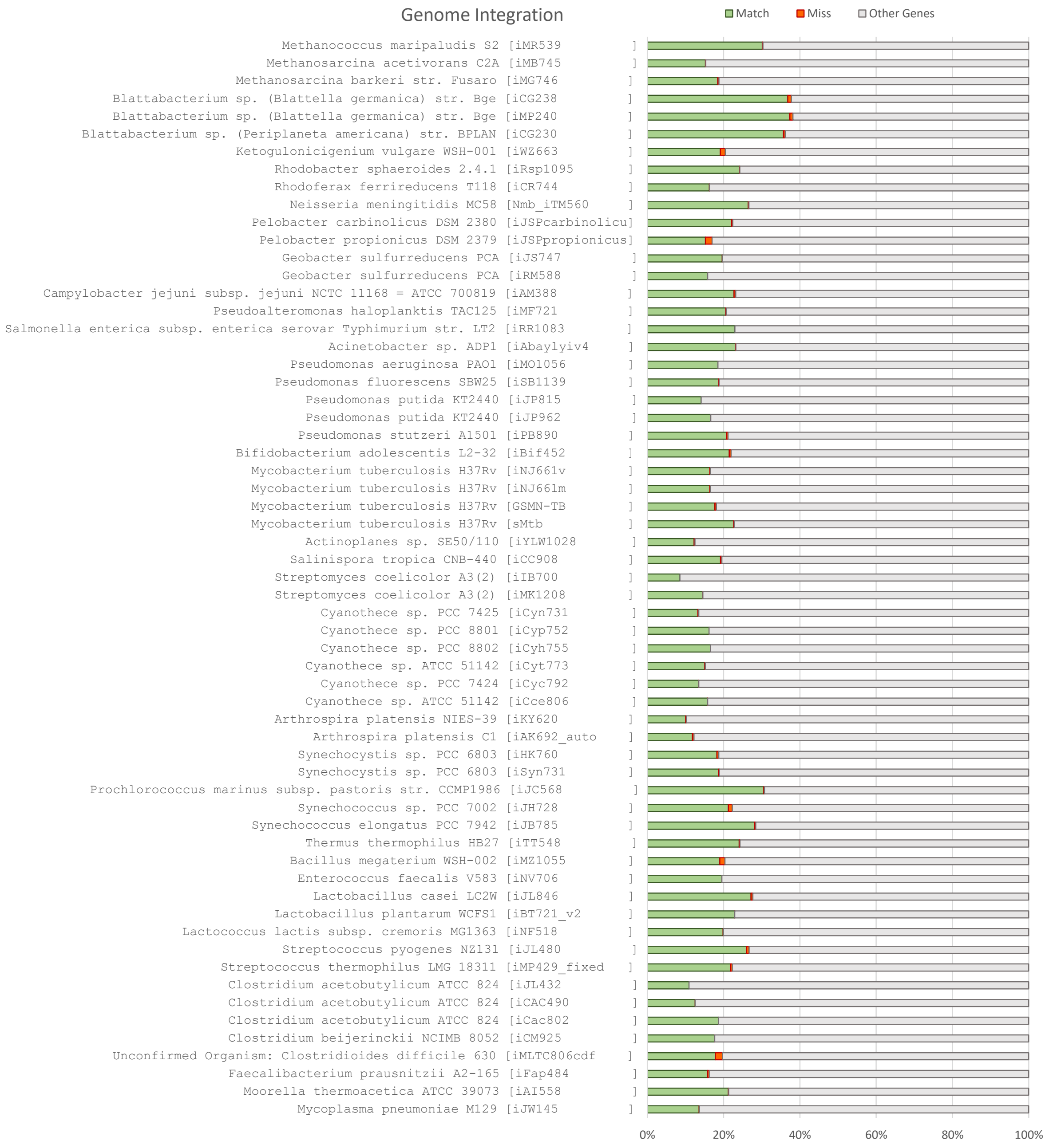


Figure 3.11: Automatic genome detection using the KBase SOLR search (highest match of all genes in the GSM). Match - genes in the GSM found in the matching genome; Miss - genes in the GSM not found in the matching genome; Other Genes - fraction of the remaining genes in the genome.

3.7 Case Study: Yeast GSM

The design of metabolic networks requires several rounds of iterations to build, refine, and test the models. Of all metabolic reconstructions, only the *Escherichia coli* and *Saccharomyces cerevisiae* had an extensive refinement from several research groups, along with a highly annotated genome due the extensive research that is conducted with these species.

There are several genome scale models available for the *Saccharomyces cerevisiae*, some inherited from previous reconstructions, others developed independently. Due to the discrepancy in the number of annotated open reading frames, and the variety of metabolites and reactions in these models, comparative analysis shows moderate divergence of the simulation results between these models. In a previous study, Heavner et al.[51] described that the OptKnock algorithm [14] would suggest different targets when different models are used, but all representing the same organism. A common problem of validating these models, is the lack of (genome-scale) experimental evidence of the internal fluxes, while most of these models are validated in a black-box fashion. The most common benchmarks to validate GSMs is to test if they are capable in predicting a certain phenotype (e.g., growth, compound utilization and production, essential genes).

The goal of this study is to create an integrated GSM space of several *Saccharomyces cerevisiae* models and extend this domain with additional information to enrich the data (e.g., additional information regarding genes, compounds, etc) for yeast research and modeling projects.

Ten of the published yeast GSM were selected for integration, most of them developed before SBML Level 3. Therefore a diverse variety of annotation and nomenclature methods were found (Table 3.2).

The iMM904, iAZ900, iND750 use the BiGG naming system, that adopts the metabolite names from the BiGG database to identify their species. Practices such as these allow to easily match the information of the metabolites with the database, without need of using annotation strategies in the SBML.

However, the BiGG database does not provide metabolite structural information, and must be obtained from other referenced databases such as KEGG or MetaCyc. The BiGG database contains only a few of the published GSMs, therefore not all metabolites may be present. In this case the iAZ900 is an external model that contains a few custom BiGG identifiers.

The consensus models of the yeast (Yeast 1, 6, 7) uses the annotation element to reference external resources.

Integration becomes complicated when there is no information available in the SBML models. This leaves the only option to analyze the annotation strategy in the SBML file to scavenge any property that may be relevant to decipher the species.

An example is the iLL672 GSM. To infer the annotation of the iLL672 GSM, the edit distance of the name strings was analyzed. The name of the species were compared with the names found in the BiGG database. Many names were similar to the names given to the BiGG compounds. This allows to rank the metabolites with minimum edit distance. But, as shown in the metabolite naming system, the name of biochemical compounds are usually ambiguous.

Most models had species identification higher than 70% (Figure 3.12). A particular case is the big difference between the annotation of the species between the Yeast 6.06 and Yeast 7.6. While the first model had a near full annotation, the Yeast 7.6 dropped to the lowest of all models. The Yeast 7.6 improved the fatty acid elongation process by unfolding the generic metabolites to the concrete representation of the fatty acids. This implies replicating the generic fatty acids pathway with each of the defined versions of the fatty acids, which increases the amount of species and reactions by many folds.

Reaction integration detected several internal duplicates across all models. The presence of enzymes that perform identical reactions but with limited reversibility must generate two reactions, as described in the previous section. In the iMM904 model, gene YMR303C encodes the reaction that converts ethanol to acetaldehyde, while the genes YOL086C, YGL256W and YBR145W perform the reverse operation. It is impossible to combine the four of these genes because they dictate different reaction constraints, however, the last three can be combined together into a disjunctive expression.

Table 3.2: Yeast metabolic models. iFF708 provides GPR in PDF supplementary files. Yeast 6/7 did not annotate subsystems for reactions. Genes - (Found in reference genome)/(Total in model)/(Total protein combinations). GPR annotation method: PDF - PDF document; XLS - Spreadsheet document; Notes - SBML notes element, Mod - SBML reaction modifiers element.

Model	Annotation	Subsystems	Species	Compartments	Reactions	Reactions (GPR)	Genes
iFF708	None	63	810	4	1382	? (PDF)	?
iLL672	None	No	672	1(3)	1194	871 (XLS)	659/660/660
iND750	BiGG	56	1177	8	1266	810 (Notes)	748/750/677
iMM904	BiGG	59	1392	8	1577	1043 (Notes)	902/905/843
iAZ900	BiGG	60	1404	8	1597	1049 (Notes)	899/901/845
iIN800	None	62	985	3	1706	1199 (Notes)	705/707/707
iTO977	XLS	105	1213	4	1562	1046 (Notes)	947/961/874
Yeast 1	RDF	60	1457	15	1857	1407 (Mod)	832/832/770
Yeast 6	RDF	?	1623	16	1888	1180 (Notes)	900/900/818
Yeast 7	RDF	?	2386	16	3493	2302 (Notes)	909/909/829

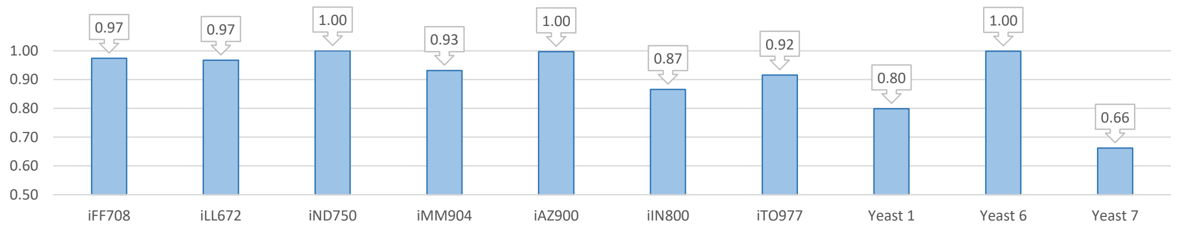


Figure 3.12: Percentage of metabolite species mapped to at least one metabolite from a database.

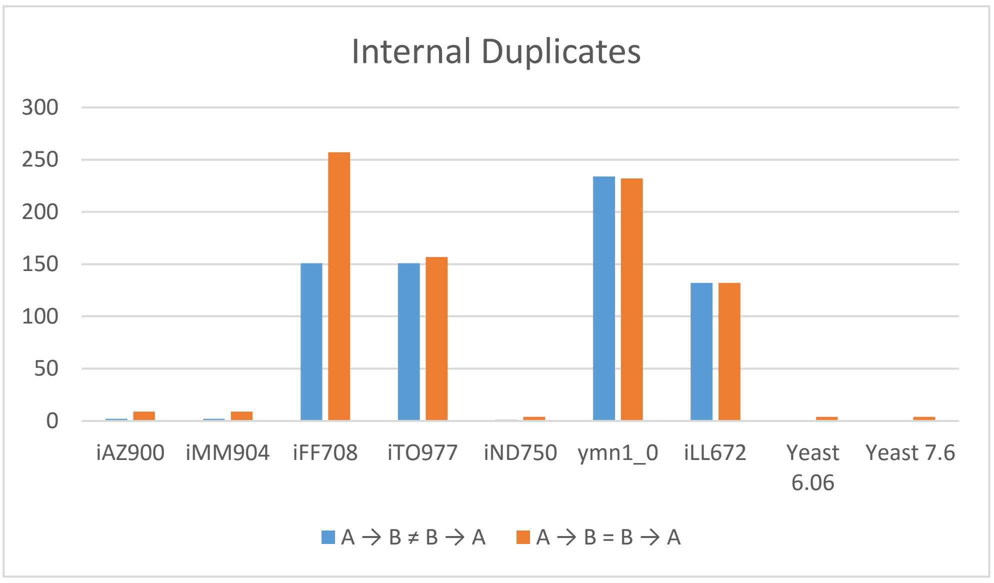


Figure 3.13: Number of duplicate reactions within each model. Blue - strict direction. Orange - ignores direction.

In some models, where the number of duplicates were much higher, it is clear that the number of reactions do not represent the actual count of distinct biochemical activities in the model. This includes the iFF708, iLL672, iTO977 and Yeast 1, that duplicates the reaction for any reversible enzyme (one for each direction).

Examples of internal duplicates were found in several GSM, such as the following occurrence in the Yeast 7.6 model:

ID	Stoichiometry	Genes Association	Bounds
r_1148:	$s\_0665 \rightarrow s\_0666$	YIL013C OR YOR011W	$]-\infty,\infty[$
r_1760:	$s\_0666 \rightarrow s\_0665$		$]-\infty,\infty[$

This is an example of possible duplicates since the flux bounds are set to be unbounded (i.e., reversible), and thus, the orientation of the components of the stoichiometry is irrelevant. By using the same comparison methods from the reaction integration, it is possible to find orientation distinct (the example above) or exact duplicates (Figure 3.13).

There are two levels of integration in the GSM. The first is to find equivalent reactions within the resources, this allows to further annotate the reactions of the GSM. Second, models are also integrated against each other. Unlike the previous case study, the yeast models are integrated against each other. Reactions not found within the metabolic databases may be shared among models, allowing to identify reactions that are unique to these models.

It is possible to observe that iAZ900 and iMM900 share a large common set of reactions, since one is inherited by the other. The Yeast 6.06 is very similar to Yeast 7.6 but the opposite is not true, since most reactions found in Yeast 6.06 are present in the Yeast 7.6. The Yeast 7.6 contains twice as many reactions of any other GSM in this study, therefore its maximum similarity is limited to a fraction compared to the remaining models.

The mapping of species plays a great deed in the integration of the stoichiometry. Models with lower coverage will eventually display poor results in the integration of the reactions. As an example, the iLL672 and iTO977 model shown low similarity. But, this is not the only reason for the lack of integration. By inspecting a few reactions, it is possible

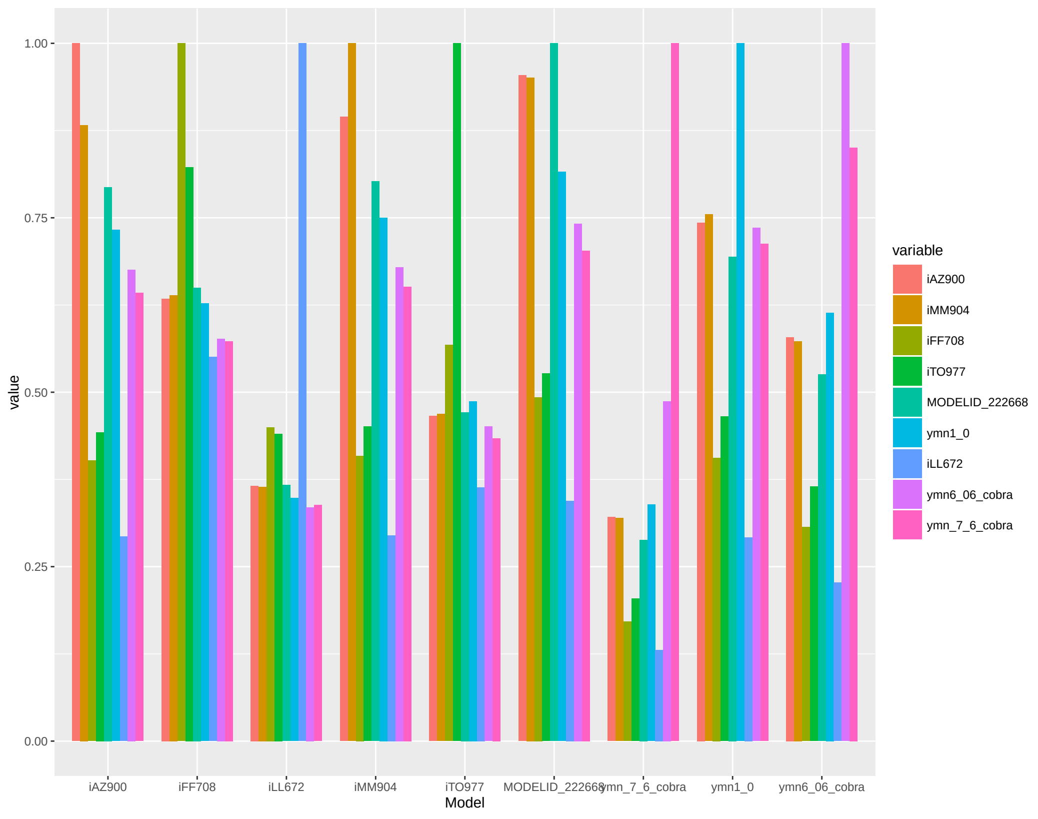


Figure 3.14: The percentage of reactions that are equal compared agains each model. MODELID_222668 - iND750; ymn6_06_cobra - Yeast 6; ymn7_7_cobra - Yeast 7

Table 3.3: Identification of the reactions in GSM with databases. Model - reactions found in another yeast models. Database - reaction found in a database. Both - reaction found in at least one model and one database.

Model	Database	Model	Any	Both	KEGG	MetaCyc	BiGG
iFF708	0.38	0.87	0.87	0.37	0.32	0.31	0.32
iLL672	0.22	0.48	0.49	0.22	0.18	0.19	0.20
iND750	0.70	0.98	0.99	0.69	0.51	0.47	0.68
iMM904	0.60	0.91	0.91	0.60	0.43	0.41	0.58
iAZ900	0.59	0.90	0.90	0.59	0.43	0.40	0.57
iIN800	0.29	0.80	0.65	0.27	0.23	0.22	0.23
iTO977	0.28	0.80	0.67	0.27	0.25	0.24	0.25
Yeast 1	0.58	0.84	0.84	0.57	0.49	0.47	0.50
Yeast 6	0.41	0.90	0.90	0.41	0.35	0.34	0.36
Yeast 7	0.21	0.50	0.50	0.21	0.18	0.18	0.18

to detect dubious stoichiometry within these models. In the iTO977 several examples of reactions with missing co-factors (e.g., R_FUM1_1, R_IPP1) were identified. The missing species from the stoichiometry of the reactions miss the integration with the remaining models.

A total of 504 genes were present in all the 9 GSMs (excludes iFF708), while a few genes were exclusive to 5 models, the iLL672, iAZ900 and Yeast 6 had 2, 2 and 3 exclusive genes. The latest model of Yeast 7 had 11 unique genes and the iTO977 included 44 genes being the model having most exclusive genes. The number of genes in the most recent models are similar, but there are small changes to the genes included (Figure 3.15), and these changes are much higher when compared to the GPRs in the models. As an example, iMM904 and iAZ900 share a similar amount of genes compared to the consensus models Yeast 1, 6, 7 (similarity index >0.85), but when it comes to proteins these numbers drop to 0.79 compared with the Yeast 7.

The biomass composition is one of the differentiation aspects of these models, from the

a)

	iFF708	iLL672	iND750	iMM904	iAZ900	iIN800	iTO977	Yeast 1	Yeast 6	Yeast 7
iFF708	1	0	0	0	0	0	0	0	0	0
iLL672	0	1	0.77	0.71	0.7	0.7	0.67	0.71	0.67	0.67
iND750	0	0.77	1	0.81	0.8	0.81	0.76	0.78	0.73	0.72
iMM904	0	0.71	0.81	1	0.99	0.75	0.87	0.9	0.88	0.87
iAZ900	0	0.7	0.8	0.99	1	0.74	0.87	0.89	0.88	0.88
iIN800	0	0.7	0.81	0.75	0.74	1	0.74	0.69	0.7	0.7
iTO977	0	0.67	0.76	0.87	0.87	0.74	1	0.8	0.85	0.85
Yeast 1	0	0.71	0.78	0.9	0.89	0.69	0.8	1	0.84	0.83
Yeast 6	0	0.67	0.73	0.88	0.88	0.7	0.85	0.84	1	0.98
Yeast 7	0	0.67	0.72	0.87	0.88	0.7	0.85	0.83	0.98	1

b)

	iFF708	iLL672	iND750	iMM904	iAZ900	iIN800	iTO977	Yeast 1	Yeast 6	Yeast 7
iFF708	1	0	0	0	0	0	0	0	0	0
iLL672	0	1	0.65	0.55	0.53	0.7	0.6	0.56	0.53	0.53
iND750	0	0.65	1	0.72	0.69	0.79	0.69	0.69	0.65	0.64
iMM904	0	0.55	0.72	1	0.95	0.67	0.74	0.87	0.81	0.78
iAZ900	0	0.53	0.69	0.95	1	0.65	0.72	0.84	0.81	0.79
iIN800	0	0.7	0.79	0.67	0.65	1	0.75	0.62	0.63	0.64
iTO977	0	0.6	0.69	0.74	0.72	0.75	1	0.69	0.73	0.73
Yeast 1	0	0.56	0.69	0.87	0.84	0.62	0.69	1	0.78	0.75
Yeast 6	0	0.53	0.65	0.81	0.81	0.63	0.73	0.78	1	0.95
Yeast 7	0	0.53	0.64	0.78	0.79	0.64	0.73	0.75	0.95	1

Figure 3.15: Comparison of genes and GPRs with Jaccard similarity index. a) - Individual genes. b) - Proteins (all *and* combinations in GPRs)

earliest model to the latest the biomass defined to yeast displayed several modifications on both coefficients and components.

The composition of biomass decides how complex the network reconstruction is. As an example, a simple biomass requires less pathways, thus a smaller (incomplete) network satisfies the growth condition.

From the yeast models a total of 11 biomass formulations were collected and tested against each other (Figure 3.16). For the components that are not compatible with the

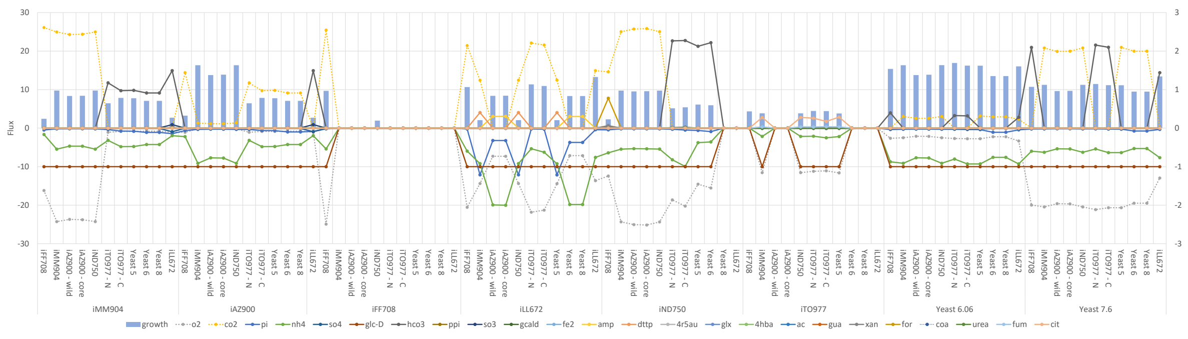


Figure 3.16: Growth rate of the yeast models by applying cross-model biomass equation. Left axis: consumption and production of compounds, Right axis (bar plots): growth rate.

model (i.e., the metabolite is not present in the model), these are discarded from the biomass.

For DNA, RNA, and protein most of the BOFs were identical except for the iLL672 and iTO977 models, that only differ in the weights.

The riboflavin was the only cofactor in the yeast biomass (except iFF708, iND750, iIN800, iTO977), only the iLL672 included other cofactors (NAD, FAD, thiamin triphosphate, coenzyme-a, tetrahydrofolate, protoheme).

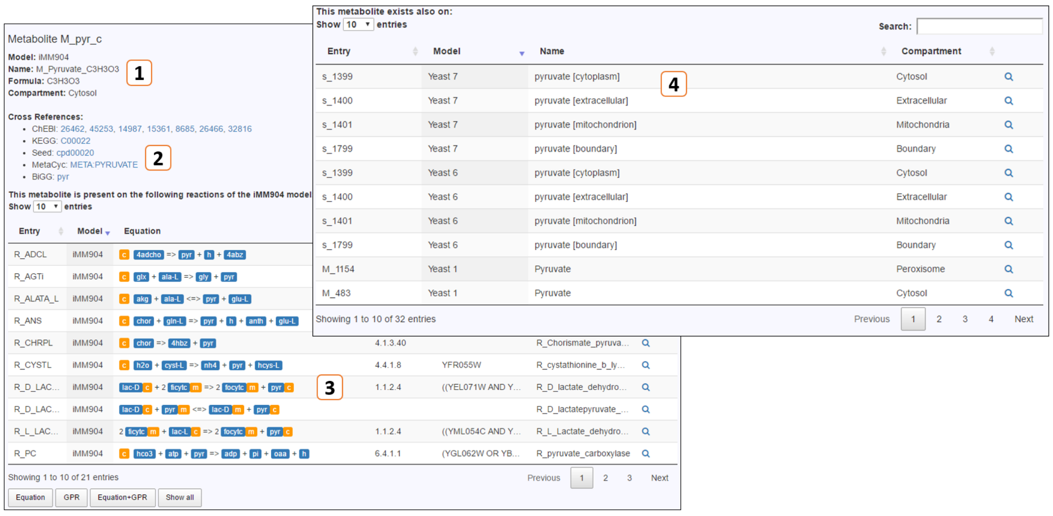
3.7.1 The Metabolic Integrated Yeast Knowledgebase

The Metabolic Integrated Yeast Knowledgebase (MIYeasTK) was created to catalog the metabolites, reactions and genes of the 10 integrated yeast models, but also integrating the genes with the Saccharomyces Genome Database (SGD) database. This work was done in collaboration with several colleagues in the research group (Christopher Costa- web interface and SGD integration, Sophia Santos - data curation) The database provides and integrated view of all the metabolites, reactions and genes of the models standardized to BiGG identifiers as aliases. Each individual component can be displayed in the integrated view that shows the occurrence of the entity in other models.

The SGD is integrated with the GPR associations of each model, transferring phenotype information from the SGD to model genes.

The compartments were standardized to the BiGG system, additional compartment aliases were created for the ones that did not exist in BiGG. Several efforts were also made to unify the pathway annotation to assign universal pathway identifiers between each of the models.

The application is implemented in Node.js, using the Neo4j as database, the REDIS in memory database is used to cache the table information to speed up filter operations. The Neo4j database is assembled from the universal graph database $\mathcal{G}$ into a dedicated version for the yeast models. The translation methods were applied to generate the BiGG aliases to the compounds for better user readability.



Metabolite M_pyr_c

Model: iMM904
Name: M_Pyruvate_C3H3O3
Formula: C3H3O3
Compartment: Cytosol

Cross References:

- ChEBI: 26462, 45253, 14987, 15361, 8685, 26466, 32816
- KEGG: C00022
- Seed: cpd00020
- MetaCyc: META:PYRUVATE
- BiGG: pyr

This metabolite is present on the following reactions of the iMM904 model
Show 10 **entries**

Entry	Model	Equation				
R_ADCL	iMM904	c 4adcho => pyr + h + 4abz				
R_AGTi	iMM904	c glx + ala-L => gly + pyr				
R_ALATA_L	iMM904	c akg + ala-L <=> pyr + glu-L				
R_ANS	iMM904	c chor + gln-L => pyr + h + anth + glu-L				
R_CHRPL	iMM904	c chor => 4hbz + pyr	4.1.3.40		R_Chorismate_pyruva...	🔍
R_CYSTL	iMM904	c h2o + cyst-L => nh4 + pyr + hcys-L	4.4.1.8	YFR055W	R_cystathionine_b_ly...	🔍
R_D_LAC...	iMM904	lac-D c + 2 ficytc m => 2 focytc m + pyr c	1.1.2.4	((YEL071W AND Y...	R_D_lactate_dehydro...	🔍
R_D_LAC...	iMM904	lac-D c + pyr m <=> lac-D m + pyr c			R_D_lactatepyruvate_...	🔍
R_L_LAC...	iMM904	2 ficytc m + lac-L c => 2 focytc m + pyr c	1.1.2.4	((YML054C AND Y...	R_L_Lactate_dehydro...	🔍
R_PC	iMM904	c hco3 + atp + pyr => adp + pi + oaa + h	6.4.1.1	(YGL062W OR YB...	R_pyruvate_carboxylase	🔍

Showing 1 to 10 of 21 entries — Previous 1 2 3 Next

Equation | GPR | Equation+GPR | Show all

This metabolite exists also on:
Show 10 **entries** — **Search:**

Entry	Model	Name	Compartment	
s_1399	Yeast 7	pyruvate [cytoplasm]	Cytosol	🔍
s_1400	Yeast 7	pyruvate [extracellular]	Extracellular	🔍
s_1401	Yeast 7	pyruvate [mitochondrion]	Mitochondria	🔍
s_1799	Yeast 7	pyruvate [boundary]	Boundary	🔍
s_1399	Yeast 6	pyruvate [cytoplasm]	Cytosol	🔍
s_1400	Yeast 6	pyruvate [extracellular]	Extracellular	🔍
s_1401	Yeast 6	pyruvate [mitochondrion]	Mitochondria	🔍
s_1799	Yeast 6	pyruvate [boundary]	Boundary	🔍
M_1154	Yeast 1	Pyruvate	Peroxisome	🔍
M_483	Yeast 1	Pyruvate	Cytosol	🔍

Showing 1 to 10 of 32 entries — Previous 1 2 3 4 Next

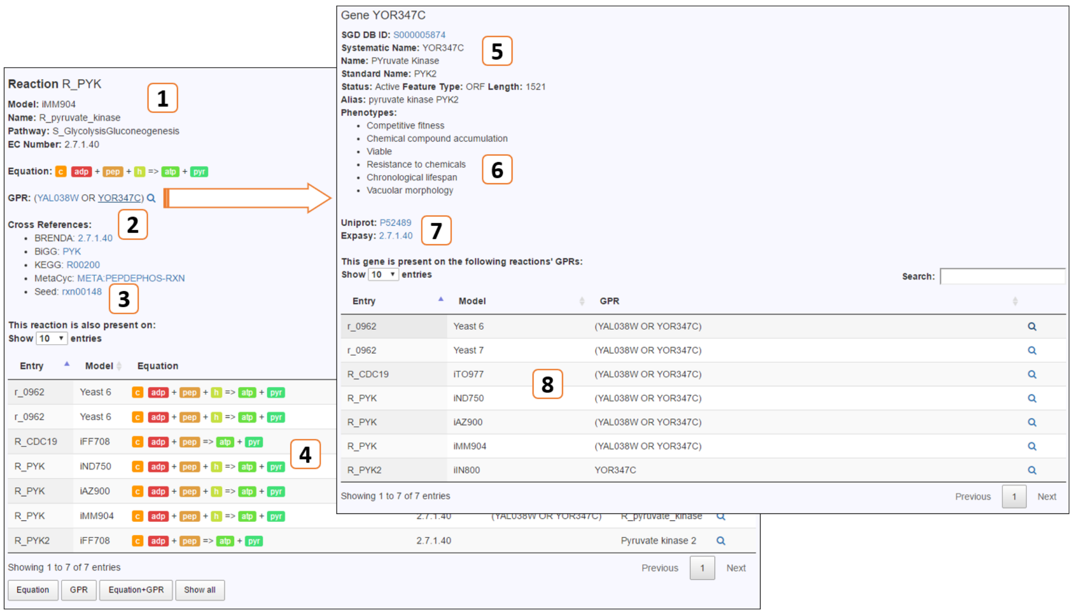
Figure 3.17: Metabolite display interface. 1 - Metabolite information. 2 - External links to metabolic databases. 3 - Reactions with the metabolite present. 4 - Other versions of the metabolite (in other compartments and models).

The metabolite display (Figure 3.17) provides external links to the model compounds, also linking reactions where the compound participates and other versions of the same compound. This offers to users a better browsing experiment to navigate between the GSM entities.

Like the compounds, the reactions (Figure 3.18) also displays links to external databases, other identical versions of the reaction across the models, and the GPR is also integrated allowing the user to browse the genes individually to explore other reactions that are associated with the gene.

3.8 Conclusions

In this chapter, many of the existing representation methods dedicated to genome-scale modeling were covered. The implemented standardization pipeline allows both reshaping the model representation, but also annotation of the model content with external database references, since they are equally relevant to enable comparative analysis of different mod-



Reaction R_PYK

Model: iMM904
Name: R_pyruvate_kinase
Pathway: S_GlycolysisGluconeogenesis
EC Number: 2.7.1.40

Equation: c adp + pep + h => atp + pyr

GPR: (YAL038W OR YOR347C)

Cross References:

- BRENDA: 2.7.1.40
- BiGG: PYK
- KEGG: R00200
- MetaCyc: META:PEPDEPHOS-RXN
- Seed: rxn00148

This reaction is also present on:
Show 10 entries

Entry	Model	Equation			
r_0962	Yeast 6	c adp + pep + h => atp + pyr			
r_0962	Yeast 6	c adp + pep + h => atp + pyr			
R_CDC19	iFF708	c adp + pep => atp + pyr			
R_PYK	iND750	c adp + pep + h => atp + pyr			
R_PYK	iAZ900	c adp + pep + h => atp + pyr			
R_PYK	iMM904	c adp + pep + h => atp + pyr	2.7.1.40	(YAL038W OR YOR347C)	R_pyruvate_kinase
R_PYK2	iFF708	c adp + pep => atp + pyr	2.7.1.40		Pyruvate kinase 2

Showing 1 to 7 of 7 entries — Previous 1 Next

Equation | GPR | Equation+GPR | Show all

Gene YOR347C

SGD DB ID: S000005874
Systematic Name: YOR347C
Name: PYruvate Kinase
Standard Name: PYK2
Status: Active **Feature Type:** ORF **Length:** 1521
Alias: pyruvate kinase PYK2
Phenotypes:

- Competitive fitness
- Chemical compound accumulation
- Viable
- Resistance to chemicals
- Chronological lifespan
- Vacuolar morphology

Uniprot: P52489
Expasy: 2.7.1.40

This gene is present on the following reactions' GPRs:
Show 10 entries — **Search:**

Entry	Model	GPR
r_0962	Yeast 6	(YAL038W OR YOR347C)
r_0962	Yeast 7	(YAL038W OR YOR347C)
R_CDC19	iTO977	(YAL038W OR YOR347C)
R_PYK	iND750	(YAL038W OR YOR347C)
R_PYK	iAZ900	(YAL038W OR YOR347C)
R_PYK	iMM904	(YAL038W OR YOR347C)
R_PYK2	iIN800	YOR347C

Showing 1 to 7 of 7 entries — Previous 1 Next

Figure 3.18: Reaction and gene display interface. 1 - Reaction information. 2 - GPR association integrated with gene display. 3 - External links to metabolic databases. 4 - Other versions of the reaction. 5 - Gene information. 6 - SGD phenotypes associated with the gene. 7 - External references to the gene. 8 - Reactions associated with the gene (across all models).

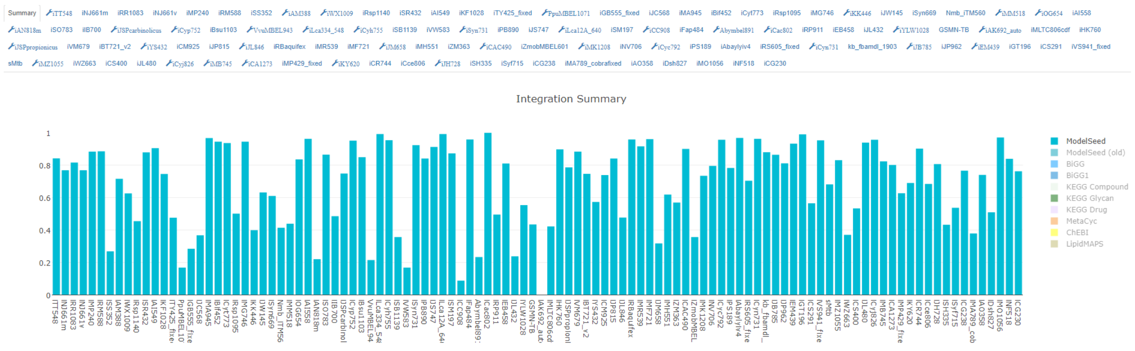


Figure 3.19: Output report for the fully automated integration of the 108 models using the implemented KBase application.

els.

The implemented application was proven to be capable to automatically annotate a SBML model with a high success rate (Figure 3.19) compared to the customized usage of the pipeline. Although there are examples where the implemented methods were unable to guess the compounds, these were quite a few (and it was only possible in the case study with the addition of data from supplementary materials). In these situations, the models usually contain user made string patterns (e.g., name concatenation with formula), being unpractical to cover each of these individual scenarios. Still, the proposed solution is flexible to the addition of new integration logic with few implications in the existing methods.

The extension of the domain logic of the CDS proven that the system is flexible to include additional domains without the need to reformulate previous implementations. This was demonstrated with the addition of SGD genes and phenotype data in the MIYeasTK database.

The integration of GSM content is data intensive, the implemented solution in the KBase platform provides the infrastructure necessary without requiring users the necessity to setup several databases. For most of the GSM that included GPR within the model, the application was able to detect the correct genome from the RefSeq repository.

The standardization tools make comparative analysis of existing GSM more practical, while it also provides a *recycling* method taking advantage of the curation efforts in the GSM models to improve future reconstruction and annotation.

Chapter 4

Pathway Optimization

Abstract

Metabolic Engineering targets the microorganism's cellular metabolism to design new strains with an industrial purpose. Applications of these metabolic manipulations in Biotechnology derive from the need of enhanced production of valuable compounds. The development of *in silico* metabolic models proposes a quantifiable approach for the manipulation of these microorganisms.

These systems are also prone to be represented as networks, taking advantage of different graph-based paradigms, including bipartite graphs, hypergraphs and process graphs. This chapter explores these representations and underlying algorithms for metabolic network topological analysis. The main aim will be to identify potential pathways towards the optimized biochemical production of selected compounds. Related to this task, algorithms will be designed aiming to complement networks of specific organisms, taking as input larger metabolic databases, inserting new reactions making them able to produce a new compound of interest.

To address these problems, and also related tasks of data pre-processing and evaluation of the solutions, a complete computational framework was developed. It inte-

grates a number of previously proposed algorithms from distinct authors, together with a number of improvements that were necessary to cope with large-scale metabolic networks. These are the result of problems identified in the previous algorithms regarding their scalability.

A case study in synthetic metabolic engineering was selected from the literature to validate the algorithms and test the capabilities of the implemented framework. It allowed to compare the performance of the implemented algorithms and validate the proposed improvements.

4.1 Pathway Optimization

4.1.1 Synthetic Pathways

An important branch of ME deals with the exploration of nonstandard routes of cellular metabolism. These routes involve metabolic activities possibly occuring in several organisms. Application of these hybrid pathways is mostly related to the synthesis of nonnative substrates in a specific microorganism, although other applications such as the study of metagenomic communities will also involve the analysis of these heterologous pathways.

The design of these pathways involves several steps. The first step is to select a proper chassis (host organism) to be used as a basis for synthesis. This chassis can be computationally represented by a metabolic network or model, if this is available for the microorganism, allowing simulation using constraint-based methods (detailed in one of the next sections).

In the following step, a suitable set of reactions must be identified that are capable to perform the synthesis of the compound of interest, from a set of source metabolites assumed to be available (produced by the host internal metabolism). For the reactions that exist in the host's metabolic portfolio, those are mapped to a set of encoding genes. For the ones that do not exist in the host, a suitable donor organism needs to be identified and the gene needs to be inserted into the genetic material of the host, using genetic engineering techniques.

From a computational point of view, the problem of finding a suitable set of reactions will be a combinatorial search to identify suitable (sub)sets of reactions from a metabolic domain, matching a specific set of constraints and, optionally, maximizing one or several criteria. This task will be referred in this work as pathway optimization, when an objective function is used to rank each set of reactions and the purpose is to maximize/ minimize it, and as pathway enumeration when the target is to enumerate all possible subsets that obey a given set of constraints.

An identical task seeks to find the pathway that enables the biodegradation of a target compound. In this scenario, the set of reactions added, instead of producing will consume the compound of interest leading to the production of compounds of interest.

While the host selection was defined as the first step, in many cases it is a problem deeply entangled with pathway optimization. Indeed, in some cases, the order of the tasks can be reversed, dealing first with the selection of interesting pathways and, afterwards selecting the best host to support these pathways. The criteria to evaluate the pathways and their interconnection with the host will be summarized in a later section of this chapter, although the complexity of this matter is out of the scope of this text.

There are several studies conducted on synthesis problems both with experimental results or just considering a computational analysis. Valuable compounds such as butanol [5], artemisinin [121], vanillin [46] or curcumin [67] were successfully engineered over several host microorganisms.

Regarding computational methods, there is a vast portfolio of algorithms available in the literature that are capable to discover and enumerate heterologous pathways. These are explained in more detail over the next sections.

4.1.2 Metabolic networks and graphs

In computational biology, Metabolic Networks (MN) play an important role being their more common and simplest representation attained by the use of regular graphs. The words network and graph are even commonly used interchangeably. In biological networks, nodes are typically biological entities such as genes, proteins or metabolites, while graph edges represent relationships between those entities (e.g. reactions in metabolism, regulatory interactions, signaling cascades). In the metabolic context, nodes are typically metabolites and edges are chemical reactions which transform metabolites, acting as substrates and products. A diversity of chemical compounds connect to each other performing chain transformations of compounds, termed as metabolic pathways.

Metabolic networks have been studied in detail and they usually exhibit typical characteristics of scale free (the majority of the nodes have very low degree, while a few nodes have a very high degree) and small world networks (average path length is smaller than what would be expected of a random network of the same size) [60]. In graph analysis methods, these properties will bring some challenges.

In ME applications, these networks are often studied for their robustness, while other analyses can be performed to understand the average diameter for each organism network and the difference between related and unrelated organisms [82].

Taking graph representations as metabolic domains, path searching algorithms can be used to extract minimal length sequences of transformations between compounds with the purpose of identifying viable pathways [26]. These make the most straightforward approaches to pathway optimization. Enumeration of possible paths is also possible, but the problem significantly grows in complexity, even considering these simple representations.

There are several studies conducted using graph paths to infer pathways from MNs. Most of these are able to enumerate pathways computing k-shortest paths between compounds. Other stochastic methods such as k-walks have also been applied to deal with the complexity of these networks. Also, the DESHARKY algorithm [108] uses Monte Carlo Markov Chains to generate solutions for both biosynthesis and biodegradation problems.

4.1.3 Limitations of regular graphs

For more complex tasks, graph representations are too simplistic since they do not fully represent the biological meaning of metabolic transformations, once many functionalities are much more complex than the information regular graphs can contain. One important example is the fact that chemical reactions are interactions involving typically more than one input and more than one output, thus not being well defined by a regular graph edge.

Related to this issue, in most scenarios, the shortest path between two compounds in a graph does not represent a biological meaningful path, since chemical reactions usually contain cofactors and pool metabolites (e.g, ATP, NAD, H_2O, H^+). The high connectivity of these compounds reroutes the shortest path (that is directly translated from a MN) to favor pool metabolites, which in most cases leads to biological meaningless solutions [36].

One solution to overcome this problem is to strip cofactors and pool metabolites (also known as currency metabolites) from the network, leaving most reactions with a single substrate and a single product. This, however, involves user expertise and manual curation of the network. Also, by removing the entire set of currency metabolites, it is impossible to obtain solutions that are able to synthesize these compounds (e.g. ATP).

An alternative is to apply weights to each compound node based on their degree [36]. Compounds with high degree are penalized, allowing shortest path methods to find the proper route avoiding currency metabolites. Nonetheless, false positives remain a problem, but compared to the previous solution, the usage of compound weights does not require chemical knowledge about the content of the network.

Additionally, graph-based systems analyzed are usually limited to linear paths over the graph. This is an important limitation since many relevant biochemical reactions have two or more substrates and/ or compounds. One solution to overcome this limitation is the implementation of further techniques to infer branched pathways over regular graphs. One of the earliest solutions is provided by the ReTrace method [103]. This algorithm involves the computation of minimal pathways, which in turn was proven to be NP-hard by reduction to the minimal set cover problem.

Apart from the mentioned solutions, these limitations have been addressed both enriching the representation, for instance with set systems, and considering additional biological information. Both approaches will be addressed in the following sections.

4.1.4 Atom Mapping

As mentioned previously, a major limitation in graph paths is the identification of the correct transitions between substrate/ products (i.e., edges) due to the presence of pool metabolites that induce short length paths that are biologically meaningless. The solutions presented previously involve the arbitrary specification of these metabolites or of arbitrary weights which may help to soften the problem, but in general do not provide a final solution. A common characteristic of graph representations of metabolic networks is that the compound itself is irrelevant for the context. In a metabolic graph, the compounds are usually presented as vertices with some unique identifier to distinguish different compounds, and the actual characteristics of the compound are all discarded (e.g. the chemical formula or structure). In this scenario, shortest path algorithms have a hard time to predict biological meaningful paths.

The atom mapping approach attacks the substrate/ product routing problem using the chemical structure of the compounds involved in the reaction. A solution to overcome this problem is to descend to a lower level where compounds are bound by atoms instead [10]. The key principle behind this approach is to track atom conservation between reactions, making it possible to infer meaningful paths between two compounds.

The application of the maximum common subgraph algorithm [3, 10] allows to track the conservation of carbon atoms between substrates and products, and therefore the conservation of carbon atoms in an entire pathway. This methodology requires no explicit assignment of currency metabolites or any heuristics to avoid these.

The KEGG Reaction pair database [72, 73] stores mapped solutions for each reaction (if available) of the KEGG reactions based on graph theoretical methods of common subgraphs of chemical structures.

This allows to bypass the requirement to atom map the reactions since they are already pre-computed. This information allows to easily access linear paths between compounds by filtering the correct transitions [35, 50].

A disadvantage of these methods is the necessity of knowing the chemical structure of the compounds. Additionally, some compounds feature generic structure (e.g., the R placeholder), and such structures can be dealt if this does not interfere with the atom conservation [50].

4.1.5 Set Systems

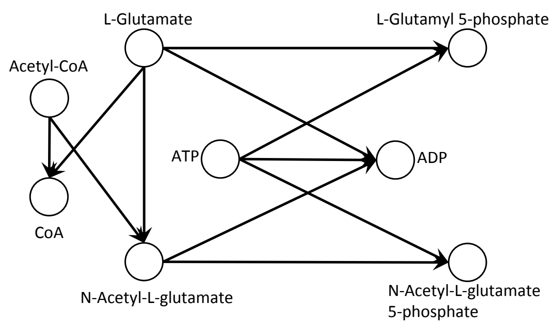
To address the limitations of regular graphs, several other models have been used to study these systems. There are several representations of graph structures depending on what level of information is captured (Figure 4.1). A common transformation is to unfold reactions into a bipartite system, such that vertices can be of two types representing compounds and reactions, while edges represent the interactions between compounds and reactions (i.e. a certain compound is a substrate or product of a reaction defining the direction of the edge) [23].

One alternative is the application of the so called set systems to represent chemical reactions, which allows to capture more complex network topologies. In these systems, entities representing reactions connect to a set of vertices instead of the binary relationships between two nodes in regular graphs. Moreover, this allows to overcome many problems related to directed graph search, for instance branching pathways and pool metabolites.

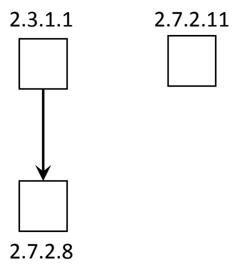
Structures such as hypergraphs [69] or process graphs [39] (which are similar to directed bipartite graphs) are set systems representations, which are capable to model chemical reactions with higher detail. This allows to address the problem of multiple products and reactants, since edges connect to vertex sets instead of a single vertex.

Process graphs were used by Friedler et al [39, 40, 41] in an exhaustive approach for decision mapping in synthesis processes, being later adapted for pathway identification [79]. More recently, the work of Carbonell et al [15] introduced an enumeration strategy to extract pathways using hypergraphs. Both algorithms are enumeration approaches that attempt to list all possible pathways towards the desired target.

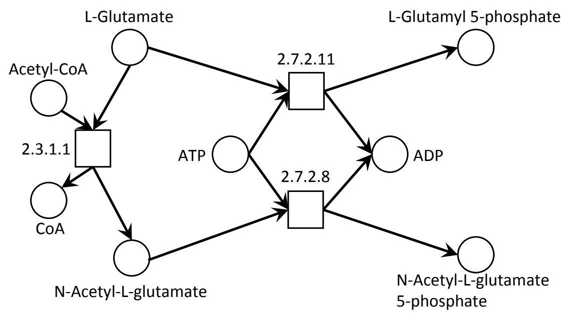
Although using these more robust structures some problems are solved, other limitations still arise. The complexity of enumerating minimal pathways was proven to be



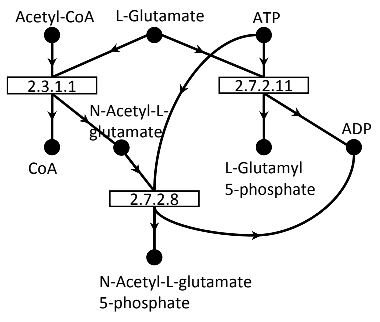
(a) Compound Graph.



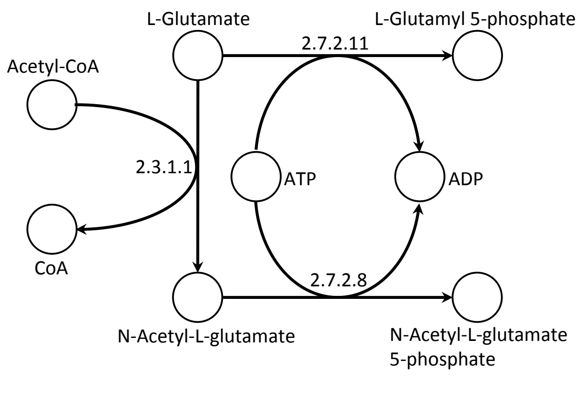
(b) Enzyme Graph



(c) Bipartite Graph



(d) Process Graph



(e) Hypergraph

Figure 4.1: Example of several network representations.

NP-hard by reduction to SAT-3 [15]. Also, the scalability of these methods is another problem to address. Finally, there are several network patterns that are not trivial to traverse, such as feedback loops.

Because of the potential of these methods, they are analyzed in further detail in the next chapter, together with pathway enumeration algorithms working over these structures. There, some of their main limitations will also be further discussed.

4.1.6 Constrain-Based Approaches

As their main advantage in pathway optimization, CBM based approaches avoid the combinatorial explosion of possible pathways in graph-based methods, through optimization based on a selected objective function. Furthermore, the constraints imposed in the system are able to guarantee that the obtained solutions are stoichiometrically valid and obey steady-state. However, a limitation is the capability to determine only a single solution and, therefore, in this regard have similar limitations to the shortest path approaches based in regular graphs. Indeed, these methods do not enumerate exhaustively other alternative solutions, which may offer valuable information on alternative routes.

A lot of effort has been put in the past years to reassemble MNs of organisms. These networks can be distinguished in two distinct categories, which are related. The term Genome Scale Network Reconstruction (GENRE) refers to a structured knowledge base for a specific network model (either a single organism or a community) [37]. These GENREs contain information that can support the creation of a mathematical model, denoted as a Genome-scale Metabolic Model (GSM). A GSM contains specific information related to the biochemistry of the model which varies in the context of the model. In CBM, these would be the constraints of the system, such as uptake boundaries for each reaction.

An an example, the KEGG database referenced earlier contains information about multiple organisms, including a network of chemical reactions most of them annotated with their corresponding enzymes and genes. These knowledge bases can be the basic building blocks to build a GSM for a specific organism [52].

4.1.6.1 Flux Balance Analysis

FBA is mainly used to conduct phenotype simulations, calculating flux distributions for different environmental and genetic conditions. By maximizing specific flux values, it can also be used to calculate the potential maximum/ minimum values for specific fluxes under the defined constraints. This is useful, for instance, to calculate maximum production values for specific compounds.

The analysis potential of the FBA is limited. The relevant information attainable from this method is the value of the objective function, while the flux distribution obtained may in some scenarios be just one of the many possible alternatives with the same optimal objective value.

FBA can be used for pathway optimization, returning a possible path (if it exists) that maximizes the used objective function. However, a MN may contain several alternative circuits to perform a certain task that could be alternative optima for FBA, or even solutions that return lower values for the defined objective function.

Nevertheless, FBA may be used as a valuable tool to be incorporated in other methods, since the computation of the optimal flux distribution is usually cheap. Among many other applications, within pathway optimization, FBA was used to determine producible non-native compounds [18] by merging a GSM with large databases such as KEGG, allowing to infer putative heterologous reactions for defined purposes.

4.1.6.2 OptStrain

In the context of synthetic pathway design, the OptStrain algorithm [101] is a CBM based approach to search for heterologous pathways, i.e. to discover a set of reactions to add to a host GSM to allow the optimal production of a non-native compound. Compared to FBA, while keeping steady-state conditions, it uses a different set of constraints to search within a domain of reactions and metabolites (assembled by the authors from the KEGG database) for the pathway with the smallest number of heterologous reactions, but with highest yield in the production of the target compound. This method is composed by four steps to design such pathways. A first step is dedicated to preprocessing and building the search domain, while the following steps are related to the optimization tasks.

Step two solves a LP to find the maximum theoretical yield. This step is important to identify if the problem is feasible and also to acquire the maximal flux value a solution may attain and is obtained with a formulation very similar to the one shown above for FBA, changing the objective function to maximize the production of the target compound.

The third step (Definition 19) involves solving a Mixed-Integer Linear Programming (MILP) problem that allows to obtain the solution that minimizes the number of non-native reactions, while keeping the production at maximal levels. The minimization of non-native reactions is obtained through reaction switches from the last three constraints (4.5, 4.6, 4.7), while the constraint (4.4) ensures the yield value $Yield^{target}$ is the maximal theoretical computed from the second task.

Definition 19. *(OptStrain) The third step of the OptStrain pipeline is defined by the following MILP:*

$$\min \quad \sum_{j \in M_{non-native}} y_j \tag{4.1}$$

$$s.t. \quad \sum_{j=1}^{M} S_{ij}.v_j \geq 0 \quad , \quad \forall i \in N, i \notin \Re \tag{4.2}$$

$$\sum_{i \in \Re} (MW_i. \sum_{j=1}^{M} S_{ij}.v_j) = -1 \quad , \tag{4.3}$$

$$MW_i. \sum_{j=1}^{M} S_{ij}.v_j \geq Yield^{target}, \quad i = P \tag{4.4}$$

$$v_j \leq v_j^{max}.y_j \quad , \quad \forall j \in M_{non-native} \tag{4.5}$$

$$v_j \leq v_j^{min}.y_j \quad , \quad \forall j \in M_{non-native} \tag{4.6}$$

$$y_j \in \{0, 1\} \quad , \quad \forall j \in M_{non-native} \tag{4.7}$$

where S_{ij} is the stoichiometric matrix with i metabolites and j reactions; v_j is flux vector; y_j are binary variable assigned to each reaction (on/ off switch); $Yield^{target}$ is the maximal theoretical flux constant; MW_i is the molecular weight of the metabolites ; P the product;

N a set of metabolites; $\Re$ a set of substrates; M a set of reactions; $M_{non-native}$ a set of reactions flagged as "non native".

After the identification of a feasible pathway it is necessary to couple to production of the product with cellular growth. This is achieved in the last step in which involves the application of the OptKnock algorithm [14] to find a set of knockouts to apply to the extended host GSM.

4.1.6.3 Elementary Flux Modes

Still within the CBM framework, Elementary Flux Modes (EFM) are defined as the minimal subsets of reactions to maintain steady state. Both previously mentioned methods are objective-based, specifying the cellular or optimization purpose. EFMs allow the enumeration of the steady state solution space given a MN. This space confines all possible routes that a steady state solution may take, which provides valuable insights of cellular capabilities and network robustness.

This space can be analyzed by considering the entire flux cone which are the extreme rays of a bounded polyhedron, which is identical to the extreme ray enumeration problem from computational geometry. The algorithms to enumerate the extreme rays are based on the Double Description method which is able to compute minimal generating sets. These minimal flux sets are sets of reactions that require the entire set to maintain the steady state constraint.

For the computation of EFMs, variants of the Double Description method are used. The canonical basis approach and the null space approach are two common variants for the computation of the EFMs, where this last algorithm is an improvement of the previous for better efficiency.

Although the EFMs offer an extreme valuable analysis tool, their complexity remains a question. Indeed, the enumeration of all EFMs in a MN is a NP-hard problem [120]. Due to this fact, the computation of EFMs is restricted to small networks [83]. Figueiredo *et al.*[25] propose an enumeration strategy to compute the k-shortest EFMs expanding the size of computable problems, but still the enumeration is computationally expensive and restricted to small values of k. Indeed, database size networks (e.g. KEGG or MetaCyc) still offer an impossible challenge for exhaustive EFM computation. For large-scale networks (e.g. GSM), the only option is to apply heuristics to reduce the search space or to use stochastic approaches [42].

4.1.7 Rule Based Systems

The other facet of MN analysis is the discovery of novel reactions or compounds based on chemical knowledge. These methods are commonly characterized as rule based approaches which share a common trait with the atom tracking approach, as they both use chemical structures to infer pathways, but with the additional capability to infer novel reactions and compounds.

A rule system applies base rules to classify reactions based on the related enzymes. A common practice is to use the Enzyme Commission (EC) classification system that involves four tiers $i.j.k.l$ classification hierarchies [48]. The i class identifies the primary function of the enzyme, while the j involves in the functional group where the enzyme acts and the remaining $k.l$ refer to the cofactors and substrates. The application of the first three tiers generates generalized enzyme rules which are not substrate specific, allowing to apply to a comprehensive range of biochemical compounds.

This approach is used in Biochemical Network Integrated Computational Explorer (BNICE) [48] framework to generate novel pathways. A reaction rule is a template reaction that transforms a certain type of compound into another. Using this approach, given a set of substrates and matching the reaction rules, it is possible to predict the products. This allows to generate a sequence of possible transformations. A rule based system allows to predict novel pathways. Such pathways can contain novel reactions that are not found in chemical databases.

The method developed by Cho *et al.*[20] predicts pathways by applying several iterations of reaction rules. The first generation loop applies all rules that match the structure of the target compound, which in turn will generate the next set of structures containing precursors of the target. After several iterations until the predefined limit, a sequence of chemical transformations are predicted that synthetize the target compound from a variety of substrates.

A major limitation of the previous methods is the high dependency on the available information of the chemical reactions. Furthermore, because of the template matching, these methods generate novel reactions which in turn, increase the number of possible combinations reaching intractable levels. Also, a higher degree of validation will be required to validate unrecorded reactions.

4.1.8 Pathway Ranking

Some of the mentioned algorithms, for pathway optimization, rely on a pre-defined objective function that internally ranks the solutions. However, the set of possible biologically meaningful criteria to rank solutions is quite vast and different application scenarios will require a distinct validation. Also, since these methods merely return computational predictions it is, in most cases, more interesting to provide an enlarged set of solutions.

Thus, we will focus here on criteria to evaluate (and rank) solutions to pathway enumeration algorithms. Indeed, graph enumeration strategies do not follow any optimization criteria, while some use the most basic topological measure: the path length. This implies that, after the computation, solutions should be scored based on biological criteria.

The ranking methods vary with the problem context. For chassis independent analysis, the size of pathway, thermodynamic feasibility and maximum achievable yield are common ranking criteria [86]. The pathway size is usually denoted by the number of reactions in the pathway. Thermodynamic feasibility is attained by computing the Gibbs free energy change of each reaction in the pathway. Lastly, the achievable yield is computed by the maximum flux value of the product divided by the flux value of the supplied carbon source (e.g., if the flux value of the product is 3 mmol/gDW/h with consumption of 2 mmol/gDW/h of glucose, then, we would obtain a yield value of 1.5).

The coupling of synthetic pathways to other information about cellular mechanisms may involve more analysis methods, in which are related to the chassis. In the DESHARKY [108] algorithm, the genetic load is also taken into consideration by calculating the energy loss in transcription and translation.

Furthermore, an important evaluation is to assess the computed pathway by integrating these in the GSM of the host microorganism. This will allow to conduct phenotype simulation to compute the *in silico* predicted performance of pathway.

The computation of ranking criteria may involve more information than the given from the optimization process (e.g., genes, enzymes and organisms that associated with the reaction). Such information may not be attainable in some scenarios due to lack of high quality curated data.

4.2 Set Systems Algorithms

In this chapter, a detailed description of the set systems algorithms (i.e., SSG and FP) is addressed. In both cases, the original algorithm will be described first, together with the limitations found. Afterwards, the proposed improvements towards better computational efficiency will be described.

Beforehand, a set of definitions is presented for a more formal definition of the synthetic metabolic problem.

4.2.1 Problem Definition

In the following, metabolic networks will be composed only by metabolites and reactions. In this system, metabolites are the vertex entities, while reactions are represented by an ordered pair $\langle M_1, M_2 \rangle$, that connects two disjoint sets of metabolites.

Definition 20. *(Reaction Simplified) A reaction is simplified to an ordered pair $\langle M_1, M_2 \rangle$ of two disjoint sets of metabolites (i.e., $M_1 \cap M_2 = \emptyset$). The first set represents the reactants, while the second represents the products.*

Definition 21. *(Metabolic Network) A metabolic network $\varSigma$ is a pair composed by a set of metabolites $\varPi$ and a set of reactions $\varUpsilon$.*

In the set systems context, the reaction is only defined with compounds. A reversible reaction r is represented by including another entity r', such that the metabolite sets are swapped. Additionally, a network $\varSigma' = \langle \varPi', \varUpsilon' \rangle$ is defined as a subnetwork of $\varSigma\langle \varPi, \varUpsilon \rangle$ if every element of $\varSigma'$ is contained in $\varSigma$ (i.e., $\varPi' \subseteq \varPi$ and $\varUpsilon' \subseteq \varUpsilon$), then $\varSigma' \subseteq \varSigma$.

Definition 22. *(Retrosyntehtic Metabolic Problem) A retrosynthetic metabolic problem $\varGamma$ is defined by a triplet $\langle \varSigma, S, T \rangle$, where $\varSigma$ is a metabolic network that represents the search space, while S and T are two disjoint sets of metabolites (i.e, $S \cap T = \emptyset$) which are the constraints of the heterologous pathways. The set S keeps the initial substrates (e.g., supplies or raw materials), while the set T defines the target compounds of interest.*

A heterologous pathway is a set of reactions, in most cases a subnetwork of a larger network (defined as the search space), that satisfies the following conditions.

Definition 23. *(Heterologous Pathway) A heterologous pathway σ of a synthetic problem $\varGamma$ is any network (or subnetwork) $\varSigma = \langle M, R \rangle$, such that: a) the product set T is included in M, i.e., $T \subset M$ and b) for every metabolite m in the subnetwork that is not included in the substrate sets of $\varGamma$ (i.e., $M - S$) there is a reaction r in R such that m is a product of r.*

The heterologous pathway definition is not sufficient to guarantee that the solution is feasible, because it omits the stoichiometry of the reactions. Both algorithms addressed in this work do not take into account this property for the computation of heterologous solutions. This eventually will lead to the computation of unfeasible solutions that later can be verified by applying FBA.

4.2.2 Solution Structure Generation

4.2.2.1 Original algorithm

The Solution Structure Generation (SSG) algorithm (shown as Algorithm 4) enumerates heterologous pathways of $\mathit{\Gamma}$ by recursively branching all possible combinations. This technique, denoted as decision mapping, can be described as follows: let $\mathit{\Sigma}'$ be a subnetwork such that condition a) in Definition 23 verifies. Then, in order to fulfill condition b), the sub-problem $\mathit{\Gamma}'$ is solved producing the unsatisfied metabolites in $\mathit{\Sigma}'$. Given for example $\mathit{\Sigma} = \langle T, \emptyset \rangle$, a network containing T and no reactions, then a) trivially verifies. Then, $\wp$(producers of t), $t \in T$ where $\wp(X)$ denotes the power set of X, are candidates for partial solutions of $\mathit{\Gamma}$, since if solutions of $\mathit{\Gamma}$ exist, at least one element of $\wp$ eventually must be present in one or more solutions of $\mathit{\Gamma}$. Recursively, we solve the sub-problem $\mathit{\Gamma}'$, with the new target set $T' = R - S - M$, where R is the set of reactants of the newly introduced reactions (minus the initial set S and producible metabolites in the partial solution), until eventually either there are no possible reactions to add (this implies that we have reached a dead end that happens when we pick a producer of T that does not belong to any solution) or $T = \emptyset$ which implies that we achieved a solution.

Algorithm 4 Solution Structure Generation

```
 1: procedure SSG($T, M, \delta[M]$)
 2:     if $T = \emptyset$ then
 3:         return $\delta[M]$                              ▷ $\delta[M]$ is a solution structure
 4:     let $x \in P$
 5:     $C \leftarrow \wp(\Delta(x)) \backslash \{\emptyset\}$            ▷ Generate all combinations of $\Delta(x)$
 6:     for $c \in C$ do                              ▷ For each combination test if is valid
 7:         if $\forall y \in m, c \cap \overline{\delta}(y) = \emptyset \wedge (\Delta(x) \backslash c) \cap \delta(y) = \emptyset$ then
 8:             $\delta[m \cup \{x\}] \leftarrow \delta[m] \cup \{(x, c)\}$
 9:             $SSG((p \cup \varphi^{-}(c)) \backslash (R \cup m \cup \{x\}), m \cup \{x\}, \delta[m \cup \{x\}])$
10:     return
```

There are several limitations of the SSG method. The first is the high amount of memory that is required to compute power sets which grow exponentially with the number of elements (2^n). Additionally, this generates an extensive amount of possible combinations. If the network is not pruned, meaning that the network contains reactions that do not belong to any solution, then the algorithm may contain branches that return no solutions and, depending on the depth of these branches, this increases severely the computation time to obtain solutions. Friedler *et al.*[40] proposed a polynomial algorithm to prune process graphs to remove all reactions that might exhibit such behavior. Because of these limitations, in the next section, we propose some modifications to the original algorithm in order to be able to compute larger networks.

4.2.2.2 Improving SSG by computing minimal solutions

The major bottleneck of the SSG algorithm, is the computation of the power set (line 6 in Algorithm 4). Furthermore, because of the union closure property of the solutions, it implies that every combination of two distinct solutions σ_α and σ_β is also a solution (i.e., $\sigma_\alpha \cup \sigma_\beta$ is a valid solution). This severely increases the amount of candidate solutions and the computation complexity of the problem.

We propose modifications to this algorithm in such way that: a) we compute only minimal solutions; and, b) we generate partitions of the power set instead of generating the entire set. A minimal solution is a solution that satisfies the steady state condition and no reaction can be removed from it. From a graph extraction viewpoint, a minimal solution implies that it cannot be disassembled into sub solutions. The condition b) allows to reach a) as it will be explained below.

Let us consider $\wp_n(X)$, which filters the power set in such way that it contains only the subsets with n elements. Then, instead of performing $C \leftarrow \wp(\Delta(x))\backslash\{\emptyset\}$, we loop through $n = 1$ to $|\Delta(x)|$, by assigning $C \leftarrow \wp_n(\Delta(x))$. This is equivalent to the line 6 of the SSG algorithm, with the advantage that we do not hold in memory the entire power set during the search.

We conjecture that, assuming a solution exists for a combination $c \in \wp_i(X)$, then every combination of higher degree $\wp_{i+1}(X)$, that contains c, can be excluded, as these do not generate the minimal solution.

Example 12. *If $X = \{a, b, c\}$ is a set with 3 elements, where $\wp(X) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$, then $\wp_0(X) = \{\emptyset\}$ is a subset of $\wp(X)$ with sets of 0 elements. Subsequently, $\wp_1(X) = \{\{a\}, \{b\}, \{c\}\}$ is the subset with all sets of 1 element and so on. Note that, for $\wp(X)$, every $\wp_n(X)$, where $n > 3$, is the empty set (i.e., $\wp_4(X) = \emptyset$).*

Given Example 12, assuming a, b, c are reactions, if we are able to find a solution for the singleton set $\{a\}$, then we exclude combinatorial sets with a (e.g., $\{a, b\}, \{a, b, c\}$). This allows to remove many, if not all, non minimal solutions thus severely increasing the capability of the SSG algorithm to perform well over larger domains.

4.2.3 Find Path

4.2.3.1 Original algorithm

The Find Path (FP) algorithm proposed by Carbonell *et al.*[15] enumerates pathways by using hypergraphs. In a metabolic context, both hypergraphs and process graphs are similar (Definition 24). A solution of the FP algorithm is defined as a *hyperpath* (Definition 25). P, which is an hypergraph (i.e., a subgraph) where the hyperarcs (reactions) can be ordered as $r_1, r_2, \ldots, r_m$, such that r_i is dependent only on the substrates in S and the products of the previous reactions.

Definition 24. *(Hypergraph) A hypergraph $\mathcal{H} = \langle V, E \rangle$ with vertices V and hyperarcs E, can be defined in this context to be isomorphic to a metabolic network Σ (Definition 21), where V represents the set of metabolites Π and E the set of reactions Υ. Additionally, a hyperarc has a structure to a reaction (Definition 20), both encompassing two disjoint sets of vertices $\langle V_1, V_2 \rangle$ (each vertex corresponds to a metabolite).*

Definition 25. *(Hyperpath [15]) A hyperpath P going from a source subset $S_{\mathcal{H}}$ of V to a target subset T_P of P in a hypergraph $\mathcal{H} = \langle V, E \rangle$ is a hypergraph $\mathcal{H}_P = \langle V_P, E_P \rangle$ with*

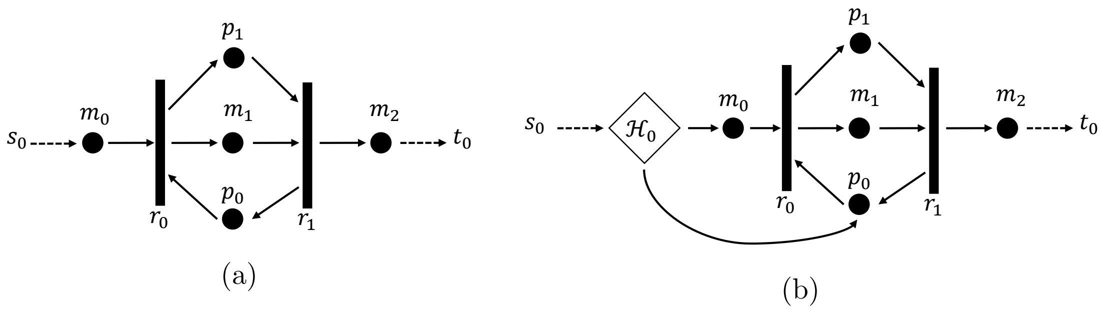


Figure 4.2: An example of a cyclic network. Vertex s_0 is the input substrate and t_0 the target metabolite. Circles represent metabolites and vertical bars represent reactions. (a) a network that does not contain pathways to produce neither p_0 nor p_1, leading to an infeasible problem to the FindPath algorithm, since no ordering is possible for reactions r_0, r_1. (b) the same network but now containing a pathway $\mathcal{H}_0$ producing p_0.

$V_P \subseteq V$, $E_P \subseteq E$, such that there is an ordering F of the hyperarcs E_P with the following properties:

- $\forall_k \in \{0, \ldots, |F|\}, substrates(F_k) \subseteq S_H \cup (\cup_{j<k} products(F_j))$
- $T_P \subseteq S_H \cup (\cup_{e_q \in E_p} products(e))$

While addressing many of the problems of using shortest paths over regular graphs to represent metabolic pathways, this representation still has limitations. Indeed, not all pathways can be expressed by the definition of an hyperpath (Definition 25). Let us consider for instance co-factor metabolites m_a and m_b.

Usually, these metabolites are both present in a single reaction $r_0 = \langle M_1, M_2 \rangle$ where $p_0 \in M_1$ and $p_1 \in M_2$ or vice versa (Figure 4.2). These reactions can be satisfied by each other in a way where there is an $r_1 = \langle M_1', M_2' \rangle$ such that $p_1 \in M_1'$ and $p_0 \in M_2'$. Therefore, it is impossible to sort a hyperpath if neither p_0 or p_1 are included in S. Given the example in Figure 4.2a, assuming $s_0 - m_0$ and $m_2 - t_0$ is feasible, then, $s_0 - t_0$ should be also feasible. But a hyperpath (Definition 25) dictates that reactions (or hyperarcs) in the hyperpath must be sortable in a particular order, where given any reaction F_k it must be satisfiable by the previous instances of $F_j, j < k$ or the initial set of substrates S_H. Now considering the two reactions r_1, r_2, this condition could never be achieved since they are dependent of each other. Examples of these metabolites are the pairs ATP-ADP and NADH-NAD. Fortunately, if assuming S to be an organism chassis (host), these metabolites are usually included in S since they are part of the metabolism. However, this does not guarantee that other more complex cycles do not exist.

This issue enables the generation of redundant solutions. Let $\Gamma = \langle \Sigma, \{s_0\}, \{t_0\} \rangle$ be a retrosynthetic problem. Assume that: a) a heterologous pathway $\Sigma' \subset \Sigma$ exists from s_0 to t_0, such that b) $r_0, r_1 \in \Sigma'$, where $r_0 = \langle \{m_0, p_0\}, \{m_1, p_1\}\} \rangle$ and $r_1 = \langle \{m_1, p_1\}, \{m_2, p_0\} \rangle$. The FP algorithm can only identify such pathway if $\Gamma' = \langle \Sigma, \{s_0\}, \{p_0, m_0\} \rangle$ is feasible. Instead of reaching from $s_0 - m_0$ as it should, the algorithm will eventually find a workaround route from $s_0 - \{m_0, p_0\}$ (Figure 4.2b). Since r_0, r_1 satisfy the metabolites p_0, p_1 of each other (i.e., $r + r' = \langle \{m_0\}, \{m_2\} \rangle$) this implies that any effort to produce p_0 in Γ' is unnecessary and every solution that b) verifies may contain multiple redundant solutions (the reactions included in the solutions are unique but in steady state they are redundant).

The Find Path algorithm (Algorithm 7) makes use of the Find All (Algorithm 5) and Minimize (Algorithm 6) subroutines. Find All (FA) implements a pruning algorithm that reduces an hypergraph $\mathcal{H}$ to $\mathcal{H}'$, with a special property: the reactions $\Upsilon \in \mathcal{H}'$ are sorted by the definition of a hyperpath. This ordering is only essential to the Find All algorithm to branch correctly, while it can be discarded (i.e., any order is acceptable) in the Minimize routine.

The Minimize routine reduces a network to the minimal set of reactions by testing each reaction in the network $\mathcal{H}$ (Algorithm 6, line 7), so that if the reaction is removed from the network, the set of products is still reachable. This testing mechanism can be achieved by invoking FA with the new network (i.e., without the reaction to be removed). If FA returns a solution without the product, then the reaction is assumed to be critical. This implies that, for each reaction in $\mathcal{H}$, an invocation of FA is performed. Therefore, the Minimize routine shows quadratic complexity to the number of reactions in the network.

Algorithm 5 Find All

```
 1: procedure FindAll($\mathcal{H}, S$)                ▷ H hypergraph, S source metabolites
 2:     for each $r \in \mathcal{H}$ do
 3:         $m[r] \leftarrow \Psi^{-}(r)$
 4:     $V \leftarrow S$
 5:     $D \leftarrow S$
 6:     $F \leftarrow \emptyset$
 7:     while $V \neq \emptyset$ do
 8:         let $x$ be an element of $V$
 9:         $V \leftarrow V \backslash x$
10:         $D \leftarrow S \cup x$
11:         for each $r \in \mathcal{H} \wedge x \in m[r]$ do
12:             $m[r] \leftarrow m[r] \backslash x$
13:             if $m[r] = \emptyset$ then
14:                 $F \leftarrow \{F, r\}$
15:                 for each $j \in \Psi^{+}(r) \wedge x \notin D$ do
16:                     $V \leftarrow V \cup j$
17:     return $F$
```

Algorithm 6 Minimize

```
1: procedure MINIMIZE($\mathcal{H}, R_f, S, T$) ▷ $\mathcal{H}$ hypergraph, $R_f$ reactions to not test, $S$ source set, $T$ target set
2:     $F \leftarrow FindAll(\mathcal{H}, S)$                                  ▷ 2-4 Test if $\mu \neq \emptyset$
3:     $\mathcal{H}' \leftarrow \mathcal{H}$
4:     if $T \cap \Psi^+(F) = \emptyset$ then
5:         $\mathcal{H}' \leftarrow \emptyset$                                  ▷ $\mu = \emptyset$ return $\emptyset$
6:     else                                          ▷ $\mu \neq \emptyset$ proceed to minimization
7:         for each $r \in H$ do      ▷ For each reaction not in $R_f$ test if $\mu \neq \emptyset$ for $\mathcal{H}\backslash r$
8:             if $r \notin R_f$ then
9:                 $F \leftarrow FindAll(\mathcal{H}\backslash r, S)$
10:                if $T \cap \Psi^+(F) \neq \emptyset$ then
11:                    $\mathcal{H}' \leftarrow \mathcal{H}'\backslash r$               ▷ Remove reaction from hypergraph
12:    return $\mathcal{H}'$                  ▷ Return either $\emptyset$ or a minimal solution structure of $\mathcal{H}$
```

4.2.3.2 Improved Minimize Heuristic

In this work, we propose an alternative to the Minimize heuristic that aims to overcome the problem of its quadratic computational complexity. We address this issue by proposing a different heuristic to test the reactions in the Minimize routine.

Assume that $\mathit{\Gamma} = \langle \mathit{\Sigma}, S, T \rangle$ contains valid solutions that are searchable using the Find Path algorithm. Assume that we increase the size of the search space to $\mathit{\Sigma}' = \langle \mathit{\Pi}', \mathit{\Upsilon}' \rangle$, where $|\mathit{\Upsilon}'|$ is much larger than $|\mathit{\Upsilon}|$. This also implies that the previous searchable solutions of $\mathit{\Gamma}$ are preserved, since it is impossible to invalidate a solution by adding more reactions to the search space. The computational cost of the previous solutions in $\mathit{\Gamma}$ will eventually increase because of: a) there are more reactions in the new network to test, therefore the computational cost of Find All increases; and, b) the Minimize now contains more reactions to remove in order to achieve the previous minimal solutions of $\mathit{\Gamma}$. Furthermore, it is natural that new solutions may be possible because of the newly added reactions in $\mathit{\Upsilon}'$.

Algorithm 7 Find Path

```
1: procedure FindPath($\mathcal{H}, R_f, S, T$)    ▷ $\mathcal{H}$ hypergraph, $S$ source metabolites, $T$ target metabolites, $R_f$ for branching solutions (initially as $\emptyset$)
2:     $F \leftarrow FindAll(\mathcal{H}, S)$
3:     $\mathcal{H}' \leftarrow \emptyset$
4:     $\mathcal{H}' \leftarrow \mathcal{H}' \cup F \cup R_f$
5:     $\mathcal{H}_\sigma \leftarrow Minimize(\mathcal{H}', R_f, S, T)$    ▷ $\mathcal{H}_\sigma$ the first minimal solution
6:     $En \leftarrow \emptyset$
7:     if $\mathcal{H}_\sigma \neq \emptyset$ then
8:         $En \leftarrow \mathcal{H}_\sigma$
9:         $F \leftarrow FindAll(\mathcal{H}_\sigma, S)$
10:        for $k \in \{|F|..1\}$ do    ▷ for each element in $F$ (i.e., hyperarcs of $\mathcal{H}_\sigma$) branch alternative solutions
11:            $r = F_k$
12:            if $r \notin R_f$ then
13:                $En \leftarrow \{En$, FindPath$(\mathcal{H} \backslash r, R_f, S, T)\}$
14:                $R_f \leftarrow R_f \cup r$
15:    return $En$
```

Our goal is to reduce the penalty to compute solutions when adding more reactions to the set. Instead of testing each reaction r (Algorithm 6, line 7), we test the removal of an entire set R of reactions. This speeds up the computation cost, specially in the search of the smallest solutions in huge networks generated from large databases, such as KEGG and MetaCyc. The size of R is an important factor, since it impacts the speed up obtained by the bulk removal of reactions.

We follow the strategy of the bisection optimization method to find the reactions that cannot be removed, thus generating a minimal set of reactions. Let X be the entire set of reactions in a network, we split X into two halves X^L and X^R, we attempt to remove from left to right each half. If X^L cannot be removed, i.e., if by removing X^L the Find All routine returns a sequence without the set T, this implies that X^L contains a reaction that must be present in the minimal solution; otherwise, there is no solution possible. Then, we split X^L into further halves X'^L, X'^R and perform again the Find All test. This routine is recursively performed until either the entire subset can be removed or we have a singleton set that cannot be removed, which implies that the reaction belongs to the minimal solution. This will generate a tree pattern where the leafs are either a singleton set with only one element (i.e., the reaction that belongs to the minimal solution) or sets of reactions that were discarded.

No modifications were made to the main Find Path algorithm.

4.3 Case Study

4.3.1 Setup

The algorithms were tested through their application to three case studies of synthetic metabolic engineering. The first example is the production of 1-butanol using *E. coli* [5], the second concerns vanillin synthesis using *S. cerevisiae* [46] and last the biosynthesis of curcumin in *E. coli*. Both modified SSG and FP algorithms are applied using the set of compounds in the KEGG Ligand and MetaCyc databases as the chemical search space.

Additionally, to integrate and test the obtained solutions *in silico*, a GSM is required: the *i*JO1366 [97] GSM for *E. coli* and *i*MM904 [88] GSM for *S. cerevisiae* were used. In both cases aerobic conditions were used with an uptake flux of glucose of 10 mmol/gDW/h. Therefore, a total of 12 result sets were generated for the two algorithms, three case studies and two search spaces (databases).

Before running the algorithms, several pre-processing tasks were required. The first was to select and define the constraints of the problem, selecting the search space Σ, the initial set S and the target compounds T. For all case studies, the target set is a singleton containing only the compound of interest (i.e., 1-butanol, vanillin and curcumin). For the substrate set, all metabolites included in the GSMMs were selected. This later will allow to integrate the obtained solutions with these models and evaluate their performance. The BiGG database [110] aided in the transformation of the species identifiers of the model to those in the databases. The species that did not match any cross-referencing were discarded.

Part of the reference pathway of the 1-butanol synthesis was mostly present in the iJO1366 GSMM as part of the Membrane Lipid Metabolism pathways. So, to obtain alternative pathways, we removed the following species: M btcoa c (Butanoyl-CoA), M btal c (Butanal), M b2coa c (Crotonyl-CoA), M 3hbcoa c (3-hydroxybutyryl-CoA), M aacoa c (Acetoacetyl-Coa). Additionally, every reaction connected to these compounds was also removed. The impact in the biomass value calculated using the FBA was minimal (less than 1%). Removing these species will allow to find alternative paths from other internal metabolites of *i*JO1366 to 1-butanol. This is done because we wanted to reach alternative solutions to the identified in [5], which may not be optimal, depending on the desired criteria. Furthermore, the algorithms do not generate solutions with reactions producing substrates in the initial set, since these are defined as supplied compounds. The curcumin case study required a new substrate in the medium, which involved the addition of a new metabolite to the *i*JO1366 GSMM, the ferulic acid.

A minor modification was made to the MetaCyc database, since it contains reactions with the metabolite pairs `NAD-P-OR-NOP`/`NADH-P-OR-NOP` which are an instance of either

`NAD/NADH` or `NADP/NADHP`. These reactions were unfolded to their correct instances. This is essential for instance to infer the 1-butanol reference pathway, as several reactions of this pathway were expressed in this format. The KEGG Ligand database did not require any pre-processing.

Both algorithms and the described modifications were implemented in Java according to the algorithms previously defined. All experiments were run on a machine running CentOS 6.4 (Linux 2.6.32) with two Intel® Xeon X5650 (2.66 GHz) and 64GBytes of memory. The java programs were compiled and run with JDK$^{\text{TM}}$7 (version 1.7.0_45). The implementation of FBA and other CBM related methods over GSMMs was taken from the core packages of the OptFlux ME platform [107] (version 3.1). The CPLEX solver (version 2.14) was used to perform the linear optimization tasks related to FBA. The KEGG information was obtained from the release 68.0 (October 1, 2013) and the MetaCyc database was taken at the same time period (release 17.5, October 11, 2013).

Because of the combinatorial explosion of possible pathways, it is impossible to obtain every solution existing in a database size network using any of the algorithms. To compare the algorithms' performance, the search space was split into subsets by *radius*. The *radius* is an integer that defines the minimum number of links (i.e., reactions) required to reach that reaction from an initial set of metabolites. This implies that a reaction belonging to radius i also belongs to $i+1$, and therefore a subnetwork Σ_i of *radius* i always complies to $\Sigma_i \subseteq \Sigma_{i+1}$.

With these reduced search spaces, solutions were computed using each of the algorithms. An attempt was made to obtain the entire set of candidate solutions for each *radius*, until either the process crashed due to lack of memory or exceeded computational time allotted ($>$ 24 hours). To validate the solutions, FBA was used to maximize the product flux of the target compound and validate its feasibility integrating the solution into the respective GSMM

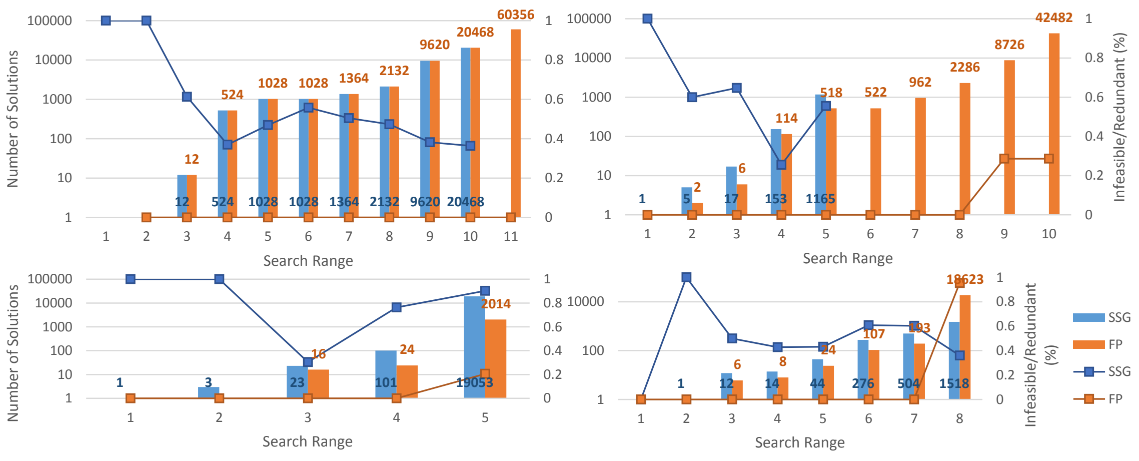


Figure 4.3: Pathways computed for each of the problems by radius.

4.3.2 Results

Figure 4.3 shows the number of solutions computed and their feasibility. SSG is more limited than FP by the size of the search space. A major problem of the SSG algorithm is the high memory demand because of the power set computation. With the reduction of the power set size (only partial sets are computed), it still presents high memory demand to branch all the possible combinations. Moreover, the SSG computes every solution that satisfies Definition 23 which eventually leads to the computation of infeasible pathways.

Still, in general, the SSG shows better performance in the computation of solutions (Figure 4.4) mainly because of the branching technique which gives a major advantage to the computation time per solution because of the backtracking. As the algorithm moves to a candidate solution, the next solution reuses the previous partial solution. This results in a neglectable impact on the computation time per solution as the search space increases (i.e., increasing size of the radius). However, since the number of solutions exponentially grows with the increasing size of the search space, the total computation time increases.

The FP is capable to compute larger search spaces, being the major bottleneck the computation time per solution, since the internal Minimize routine has quadratic complexity to the number of reactions [15]. A scenario was also found where FP computes multiple distinct redundant solutions, due to the problems explained above in detail.

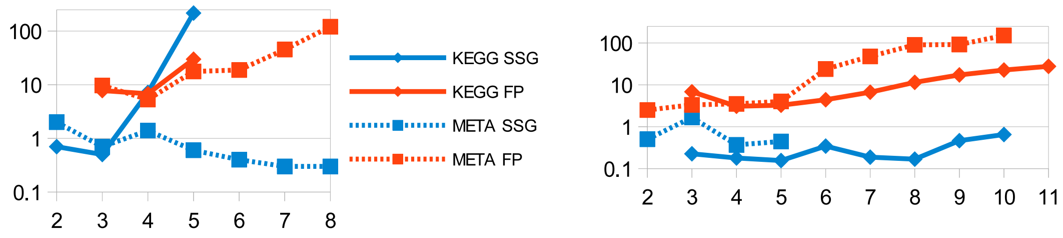


Figure 4.4: Time cost (milliseconds) per each solution. On the x-axis is the search radius (a higher radius implies a larger search space).

Table 4.1: Number of solutions obtained for the curcumin case study (on the left the number of solutions feasible with the iJO1366 GSM). For the KEGG dataset, solutions are up to radius 5 and 3 for FP and SSG, respectively. The MetaCyc dataset was fully computed.

	FindPath		SSG	
	Total	Feasible	Total	Feasible
KEGG	285	217	5	5
MetaCyc	10	7	10	7

The curcumin case study revealed a much smaller solution diversity (Table 4.1) since the amount of solutions is highly dependent on the diversity of reactions in the search space. Curcumin is a compound found originally in a few plants and thus the diversity of pathways for its production is still low. Both SSG and FP were able to fully compute the entire dataset of reactions in MetaCyc obtaining just a few solutions. The FP method was able to compute a much higher amount of solutions using the KEGG reaction set; however the SSG was unable to pass the 4th radius having only five solutions in the 3rd radius of the KEGG search space. The KEGG dataset showed increased complexity compared to the MetaCyc reactions which led the SSG algorithm to block due to memory limitations. Again the FP algorithm prove to be more capable of obtaining complex pathways mostly due to the assumption that pathways are acyclic.

For every solution that satisfies the feasibility test, the fitness was evaluated by integrating it into the corresponding GSM. The farthest *radius* that either algorithm was able to compute was selected for this process. For the 1-butanol case, from the 42482 and 60356 solutions obtained from the FP algorithm, a total of 32692 and 22968 were compatible with the iJO1366 GSM for search spaces of MetaCyc and KEGG, respectively. In the vanillin case, 944 out of 974 computed solutions are valid (MetaCyc), being the numbers for KEGG of 1600 out of 1852. Finally, for the curcumin pathways 217 out of 285 KEGG pathways and 7 out of 10 MetaCyc pathways were feasible with the iJO1366 GSM. The 1-butanol case shown a massive amount of solutions mostly because of the NAD/NADH alternatives for many reactions.

The KEGG dataset provided the solution with highest yield for vanillin and curcumin. Moreover, 152 pathways were found in KEGG with the maximum yield for 1-butanol (0.99, given by 9.99 mmol/gDW/h for the butanol production flux divided by 10 mmol/gDW/h for glucose uptake) compared to 114 pathways from MetaCyc, while for the curcumin case study the amount of solutions obtained from MetaCyc is quite limited. There is a noticeable difference in the configuration of the yield distribution between KEGG and MetaCyc (Figure 4.5), which demonstrates that there are key reactions that are unique to each database, therefore leading to different pathway configurations.

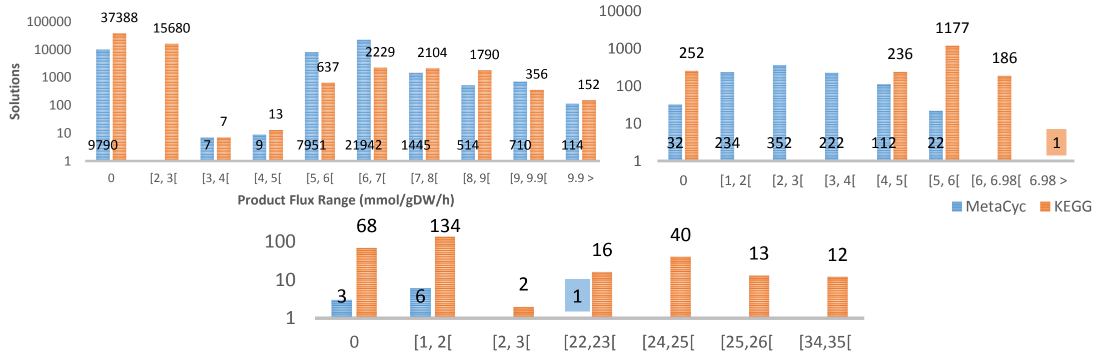


Figure 4.5: Histogram of theoretical flux values of each case study (1-butanol/curcumin - iJO1366; vanillin - *i*MM904). Last value is the optimal solution (for better product flux).

In summary, it can be concluded that overall the algorithms were able to find widely known efficient pathways but also less utilized ones. For example, in the case of butanol, the best performing pathways in terms of yield include the commonly used pathway from *Clostridium acetobutylicum*, which has also been validated [5] as a heterologous pathway in *E. coli*, but also less common pathways that have been recently patented and that use 2-ketoisovalerate as an intermediate [126]. Moreover, pathways that use amino-acids as precursors have also been identified, such as the one recently described which starts from glycine [11]. In the case of curcumin, most of the solutions take tyrosine as a precursor, as has been described elsewhere [67]. Nevertheless, in both cases there are many alternatives that are stoichiometrically feasible but for which no reports have been found in the literature. Those cases need to be further inspected for biological and biochemical consistency before implementation. Nevertheless, they constitute promising alternatives to produce valuable products.

4.4 Conclusions

The algorithms analyzed (SSG and FP) both present shortcomings in the computation of heterologous pathways. Although topologically they are correct, they may be stoichiometrically inconsistent within a microorganism's context, as they have the common goal of inferring heterologous pathways (subnetworks) that satisfy the rules of initial substrates and target product. However, by using post-processing methods such as FBA, stoichiometrically valid solutions can be identified, which allows to correctly enumerate multiple steady-state pathways. The case study of 1-butanol shows that there are many viable and optimally efficient (regarding yields) routes for the production of this compound using as basis the iJO1366 model. Moreover, even if a problem contains only a single optimal solution (e.g., vanillin in iMM904), examples of sub-optimal pathways also show a broad range of yield value near the optimal. Due to their nature, deterministic methods hardly can achieve such a range of feasible steady state heterologous pathways.

Overall, the FP has proven to be more flexible regarding the complexity and the size of the graph, and, although being more penalized with the number of reactions in the search space, it is more capable to compute larger sets.

Thus, it is shown that although neither of the algorithms is readily suitable to compute steady state heterologous pathways for large databases, they are still able extract potential pathways, after targeted improvements in scalability. Additionally, they offer a generic method to infer pathways for multiple purposes, since they do not follow any strict objective function (e.g., yield or size).

As future work, both these algorithms can still be improved towards their scalability. One line of work will certainly be the efficient parallelization of these algorithms resorting to adequate software development tools [102]. A complementary research topic will address the comparison of these approaches with recent proposals within EFM research.

Chapter 5

Conclusions

Genome-scale metabolic modeling is highly data oriented, and the completeness of these models is many times dependent on the existing information to describe either directly or indirectly the biological mechanism.

Databases are often specialized into a certain topic, and their integration allows to gather all the individual strengths into a single location for better decision making.

Integration errors are inevitable because of many reasons (poorly described data, ambiguous definitions, etc). In existing integrated databases, it is difficult to evaluate which portion of the information is most reliable. The proposed methods allow to control the certainty of the integration, allowing users to choose which configuration suits best their needs.

Demanding a high degree confidence level to the integration may generate less clusters. but it scaffolds a high confidence initial set. The implemented pipeline allows to combine integrations (because of the curation function), which permits to generate consensus sets by combining the solution of previous or external integration.

Even though SBML provides a structure to represent GSM entities, the flexibility of the annotation methods (i.e., both notes and annotation elements) allowed developers to adopt their own strategies to fit additional data into the models. The 108 models used as benchmark showed that it is possible to automatically annotate models with a high success rate. Still in a few cases, it is also shown that without user intervention, it is unlikely that there is any chance for automated annotation.

The centralized database provided a rich dataset of metabolite references and names increasing the chance for annotation. Compound names proven to be essential to integrate metabolites since genome-scale models usually inherit attributes from the databases.

Integrating compounds, reactions and genes is data intensive. However, the KBase platform provides these resources saving tremendous effort that otherwise would require the user to setup. With the entire prokaryote genome catalog of RefSeq, the integration application was able to automatically integrate the genome features with the genes found in the GPRs in most of the models, thus allowing to automatically detect the correct species/strain genome for a given model. In average, most of the examples when methods failed to identify the correct instances, it implied custom string patterns that were introduced in the models (e.g., concatenation of formula with the name attribute, modified gene names).

The integration of GSM makes models compatible with each other making comparative analysis studies more viable but also more scalable. Each of these published models contains many curation efforts to assign metabolic functions to the organism's genome. The unification of these models enables large scale knowledge extraction to reuse their information for future reconstructions.

The complete enumeration of minimal pathways of the entire metabolic space of databases is unlikely to be possible because of the combinatorial explosion. However, fully enumeration is possible by limiting the size of the pathways this would at least exhaust all possible solutions within its range. The improvements made to the enumeration methods allow to scale the methods to much larger problem sizes shifting the bottleneck to memory instead of computational power.

5.1 Main Contributions

Metabolic Database Integration Pipeline: The integration pipeline follows a flexible and configurable approach to unify metabolic databases. The main concern of the pipeline is to provide a maintainable strategy to have a flexible system that facilitates the synchronization of the reference spaces of the metabolic databases. The high parameterization of the integration methods allows for a better control of the confidence scores of the given solutions.

SBML Standardization Application: The module developed for the KBase platform allows to integrate external SBML models with the KBase system. Since a SBML model is the only required input data, the application offers a practical approach for any user to annotate existing models but also to export the matching genome and default simulation constraints (media).

Minimal Pathway Enumeration Methods: Fully enumeration can be achieved using graph methods up to a certain size limit of the pathways. The improved searching kernels allows to increase this limit to a much larger size.

5.2 Future Perspectives

Hierarchy and Ontology: The relationship between compounds is hierarchical, in many databases this hierarchy is not detailed. Extending the integration to include hierarchy would greatly benefit the integrated solution since it would allow to create additional relationship between databases.

Model Reconstruction: The standardization of models allows to extract curated knowledge about the metabolic features of the organisms. Using existing models to propagate previous models is a common reconstruction strategy. With the increasing number of curated models, it provides a rich dataset of annotated metabolic networks of organisms to aid future reconstructions.

Integrated Pathway Optimization: Current methods are oriented to find optimal routes targeted to a single organism. With standardized models it is possible to extend the methods to optimization against a set of organisms.

Bibliography

[1] R. Agren, L. Liu, S. Shoaie, W. Vongsangnak, I. Nookaew, and J. Nielsen. The RAVEN toolbox and its use for generating a genome-scale metabolic model for Penicillium chrysogenum. *PLoS Computational Biology*, 9(3):e1002980, jan 2013.

[2] A. V. Aho and M. J. Corasick. Efficient string matching: an aid to bibliographic search. *Communications of the ACM*, 18(6):333–340, jun 1975.

[3] M. Arita. In silico atomic tracing by substrate-product relationships in Escherichia coli intermediary metabolism. *Genome research*, 13(11):2455–66, nov 2003.

[4] M. Ataman and V. Hatzimanikatis. Heading in the right direction: thermodynamics-based network analysis and pathway engineering. *Current Opinion in Biotechnology*, 36:176–182, dec 2015.

[5] S. Atsumi, A. F. Cann, M. R. Connor, C. R. Shen, K. M. Smith, M. P. Brynildsen, K. J. Y. Chou, T. Hanai, and J. C. Liao. Metabolic engineering of Escherichia coli for 1-butanol production. *Metabolic engineering*, 10(6):305–11, nov 2008.

[6] R. K. Aziz, D. Bartels, A. A. Best, M. DeJongh, T. Disz, R. A. Edwards, K. Formsma, S. Gerdes, E. M. Glass, M. Kubal, F. Meyer, G. J. Olsen, R. Olson, A. L. Osterman, R. A. Overbeek, L. K. McNeil, D. Paarmann, T. Paczian, B. Parrello, G. D. Pusch, C. Reich, R. Stevens, O. Vassieva, V. Vonstein, A. Wilke, and O. Zagnitko. The RAST Server: Rapid Annotations using Subsystems Technology. *BMC Genomics*, 9(1):75, jan 2008.

[7] R. K. Aziz, S. Devoid, T. Disz, R. A. Edwards, C. S. Henry, G. J. Olsen, R. Olson, R. A. Overbeek, B. Parrello, G. D. Pusch, R. L. Stevens, V. Vonstein, and F. Xia. SEED servers: High-Performance Access to the SEED Genomes, Annotations, and Metabolic Models. *PloS one*, 7(10):e48053, jan 2012.

[8] S. L. Bell and B. Ø. Palsson. Expa: a program for calculating extreme pathways in biochemical reaction networks. *Bioinformatics*, 21(8):1739–1740, apr 2005.

[9] T. Bernard, A. Bridge, A. Morgat, S. Moretti, I. Xenarios, and M. Pagni. Reconciliation of metabolites and biochemical reactions for metabolic networks. *Briefings in Bioinformatics*, 15(1):123–135, jan 2014.

[10] F. Boyer and A. Viari. Ab initio reconstruction of metabolic pathways. *Bioinformatics*, 19(Suppl 2):ii26–ii34, oct 2003.

[11] P. Branduardi, V. Longo, N. M. Berterame, G. Rossi, and D. Porro. A novel pathway to produce butanol and isobutanol in Saccharomyces cerevisiae. *Biotechnology for biofuels*, 6(1):68, jan 2013.

[12] E. Brunk, N. Mih, J. Monk, Z. Zhang, E. J. O'Brien, S. E. Bliven, K. Chen, R. L. Chang, P. E. Bourne, and B. Ø. Palsson. Systems biology of the structural proteome. *BMC Systems Biology*, 10(1):26, dec 2016.

[13] P. Burek, R. Hoehndorf, F. Loebe, J. Visagie, H. Herre, and J. Kelso. A top-level ontology of functions and its application in the Open Biomedical Ontologies. *Bioinformatics*, 22(14):e66–e73, jul 2006.

[14] A. P. Burgard, P. Pharkya, and C. D. Maranas. Optknock: a bilevel programming framework for identifying gene knockout strategies for microbial strain optimization. *Biotechnology and bioengineering*, 84(6):647–57, dec 2003.

[15] P. Carbonell, D. Fichera, S. B. Pandit, and J.-L. Faulon. Enumerating metabolic pathways for the production of heterologous target chemicals in chassis organisms. *BMC systems biology*, 6(1):10, jan 2012.

[16] R. Caspi, R. Billington, L. Ferrer, H. Foerster, C. A. Fulcher, I. M. Keseler, A. Kothari, M. Krummenacker, M. Latendresse, L. A. Mueller, Q. Ong, S. Paley, P. Subhraveti, D. S. Weaver, and P. D. Karp. The MetaCyc database of metabolic pathways and enzymes and the BioCyc collection of pathway/genome databases. *Nucleic Acids Research*, 44(D1):D471–D480, jan 2016.

[17] J. Chambers, M. Davies, A. Gaulton, A. Hersey, S. Velankar, R. Petryszak, J. Hastings, L. Bellis, S. McGlinchey, and J. P. Overington. UniChem: a unified chemical structure cross-referencing and identifier tracking system. *Journal of cheminformatics*, 5(1):3, jan 2013.

[18] S. Chatsurachai, C. Furusawa, and H. Shimizu. An in silico platform for the design of heterologous pathways in nonnative metabolite production. *BMC Bioinformatics*, 13(1):93, jan 2012.

[19] L. Chindelevitch, J. Trigg, A. Regev, and B. Berger. An exact arithmetic toolbox for a consistent and reproducible structural analysis of metabolic network models. *Nature communications*, 5:4893, jan 2014.

[20] A. Cho, H. Yun, J. H. Park, S. Y. Lee, and S. Park. Prediction of novel synthetic pathways for the production of desired chemicals. *BMC Systems Biology*, 4:35, jan 2010.

[21] W. B. Copeland, B. A. Bartley, D. Chandran, M. Galdzicki, K. H. Kim, S. C. Sleight, C. D. Maranas, and H. M. Sauro. Computational tools for metabolic engineering. *Metabolic Engineering*, 14(3):270–280, may 2012.

[22] M. Courtot, N. Juty, C. Knupfer, D. Waltemath, A. Zhukova, A. Drager, M. Dumontier, A. Finney, M. Golebiewski, J. Hastings, S. Hoops, S. Keating, D. B. Kell, S. Kerrien, J. Lawson, A. Lister, J. Lu, R. Machne, P. Mendes, M. Pocock, N. Rodriguez, A. Villeger, D. J. Wilkinson, S. Wimalaratne, C. Laibe, M. Hucka, and

N. Le Novere. Controlled vocabularies and semantics in systems biology. *Molecular Systems Biology*, 7(1):543–543, apr 2014.

[23] D. Croes, F. Couche, S. J. Wodak, and J. van Helden. Metabolic PathFinding: inferring relevant pathways in biochemical networks. *Nucleic Acids Research*, 33(Web Server issue):W326–30, jul 2005.

[24] D. A. Cuevas, J. N. Edirisinghe, C. S. Henry, R. Overbeek, T. G. O'Connell, and R. A. Edwards. From DNA to FBA: How to Build Your Own Genome-Scale Metabolic Model. *Frontiers in Microbiology*, 7(JUN):1–12, jun 2016.

[25] L. F. de Figueiredo, A. Podhorski, A. Rubio, C. Kaleta, J. E. Beasley, S. Schuster, and F. J. Planes. Computing the shortest elementary flux modes in genome-scale metabolic networks. *Bioinformatics (Oxford, England)*, 25(23):3158–65, dec 2009.

[26] Y. Deville, D. Gilbert, J. van Helden, and S. J. Wodak. An overview of data models for the analysis of biochemical pathways. *Briefings in bioinformatics*, 4(3):246–59, sep 2003.

[27] O. Dias, D. Gomes, P. Vilaca, J. Cardoso, M. Rocha, E. C. Ferreira, and I. Rocha. Genome-Wide Semi-Automated Annotation of Transporter Systems. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 14(2):443–456, mar 2017.

[28] O. Dias, M. Rocha, E. C. Ferreira, and I. Rocha. Merlin: Metabolic Models Reconstruction Using Genome-Scale Information. In *Computer Applications in Biotechnology*, number Cab, pages 120–125, 2010.

[29] O. Dias, M. Rocha, E. C. Ferreira, and I. Rocha. Reconstructing genome-scale metabolic models with merlin. *Nucleic Acids Research*, 43(8):3899–3910, apr 2015.

[30] K. Dolinski and O. G. Troyanskaya. Implications of Big Data for cell biology. *Molecular Biology of the Cell*, 26(14):2575–2578, jul 2015.

[31] A. Ebrahim, E. Almaas, E. Bauer, A. Bordbar, A. P. Burgard, R. L. Chang, A. Drager, I. Famili, A. M. Feist, R. M. Fleming, S. S. Fong, V. Hatzimanikatis, M. J. Herrgård, A. Holder, M. Hucka, D. Hyduke, N. Jamshidi, S. Y. Lee, N. Le Novere, J. A. Lerman, N. E. Lewis, D. Ma, R. Mahadevan, C. D. Maranas, H. Nagarajan, A. Navid, J. Nielsen, L. K. Nielsen, J. Nogales, A. Noronha, C. Pal, B. Ø. Palsson, J. A. Papin, K. R. Patil, N. D. Price, J. L. Reed, M. Saunders, R. S. Senger, N. Sonnenschein, Y. Sun, and I. Thiele. Do genome-scale models need exact solvers or clearer standards? *Molecular Systems Biology*, 11(10):831–831, oct 2015.

[32] A. Ebrahim, J. A. Lerman, B. Ø. Palsson, and D. R. Hyduke. COBRApy: COnstraints-Based Reconstruction and Analysis for Python. *BMC Systems Biology*, 7(1):74, 2013.

[33] I. Famili and B. Ø. Palsson. The convex basis of the Left Null Space of the Stoichiometric Matrix Leads to the definition of metabolically meaningful pools. *Biophysical Journal*, 85(1):16–26, jul 2003.

[34] J. P. Faria, R. Overbeek, R. C. Taylor, N. Conrad, V. Vonstein, A. Goelzer, V. Fromion, M. Rocha, I. Rocha, and C. S. Henry. Reconstruction of the Regulatory Network for Bacillus subtilis and Reconciliation with Gene Expression Data. *Frontiers in Microbiology*, 7(MAR):1–11, mar 2016.

[35] K. Faust, D. Croes, and J. van Helden. Metabolic pathfinding using RPAIR annotation. *Journal of molecular biology*, 388(2):390–414, may 2009.

[36] K. Faust, P. Dupont, J. Callut, and J. van Helden. Pathway discovery in metabolic networks by subgraph extraction. *Bioinformatics (Oxford, England)*, 26(9):1211–8, may 2010.

[37] A. M. Feist, M. J. Herrgård, I. Thiele, J. L. Reed, and B. Ø. Palsson. Reconstruction of biochemical networks in microorganisms. *Nature Review Microbiology*, 7:129–143, 2009.

[38] A. M. Feist and B. Ø. Palsson. The biomass objective function. *Current Opinion in Microbiology*, 13(3):344–349, jun 2010.

[39] F. Friedler, K. Tarján, Y. Huang, and L. T. Fan. Graph-theoretic approach to process synthesis: axioms and theorems. *Chemical Engineering Science*, 47(8):1973–1988, jun 1992.

[40] F. Friedler, K. Tarjan, Y. W. Huang, and L. T. Fan. Graph-theoretic approach to process synthesis: Polynomial algorithm for maximal structure generation. *Computers & Chemical Engineering*, 17(9):929–942, sep 1993.

[41] F. Friedler, J. Varga, and L. T. Fan. Decision-mapping: A tool for consistent and complete decisions in process synthesis. *Chemical Engineering Science*, 50(11):1755–1768, jun 1995.

[42] J. Gebauer, S. Schuster, L. F. de Figueiredo, and C. Kaleta. Detecting and investigating substrate cycles in a genome-scale human metabolic network. *The FEBS journal*, 279(17):3192–202, sep 2012.

[43] C. Goble and R. Stevens. State of the nation in data integration for bioinformatics. *Journal of Biomedical Informatics*, 41(5):687–693, oct 2008.

[44] D. Gomez-Cabrero, I. Abugessaisa, D. Maier, A. Teschendorff, M. Merkenschlager, A. Gisel, E. Ballestar, E. Bongcam-Rudloff, A. Conesa, and J. Tegnér. Data integration in the era of omics: current and future challenges. *BMC Systems Biology*, 8(Suppl 2):I1, 2014.

[45] L. Haas. Beauty and the Beast: The Theory and Practice of Information Integration. chapter Beauty and, pages 28–43. 2006.

[46] E. H. Hansen, B. L. Møller, G. R. Kock, C. M. Bünner, C. Kristensen, O. R. Jensen, F. T. Okkels, C. E. Olsen, M. S. Motawia, and J. Hansen. De Novo Biosynthesis of

Vanillin in Fission Yeast (Schizosaccharomyces pombe) and Baker's Yeast (Saccharomyces cerevisiae). *Applied and Environmental Microbiology*, 75(9):2765–74, may 2009.

[47] J. Hastings, P. de Matos, A. Dekker, M. Ennis, B. Harsha, N. Kale, V. Muthukrishnan, G. Owen, S. Turner, M. Williams, and C. Steinbeck. The ChEBI reference database and ontology for biologically relevant chemistry: enhancements for 2013. *Nucleic acids research*, 41(Database issue):D456–63, jan 2013.

[48] V. Hatzimanikatis, C. Li, J. a. Ionita, C. S. Henry, M. D. Jankowski, and L. J. Broadbelt. Exploring the diversity of complex metabolic networks. *Bioinformatics (Oxford, England)*, 21(8):1603–9, apr 2005.

[49] C. T. Have and L. J. Jensen. Are graph databases ready for bioinformatics? *Bioinformatics*, 29(24):3107–3108, dec 2013.

[50] A. P. Heath, G. N. Bennett, and L. E. Kavraki. Finding metabolic pathways using atom tracking. *Bioinformatics (Oxford, England)*, 26(12):1548–55, jun 2010.

[51] B. D. Heavner and N. D. Price. Comparative Analysis of Yeast Metabolic Network Models Highlights Progress, Opportunities for Metabolic Reconstruction. *PLOS Computational Biology*, 11(11):e1004530, nov 2015.

[52] B. D. Heavner, K. Smallbone, B. Barker, P. Mendes, and L. P. Walker. Yeast 5 - an expanded reconstruction of the Saccharomyces cerevisiae metabolic network. *BMC Systems Biology*, 6(1):55, jan 2012.

[53] S. Heller, A. McNaught, S. Stein, D. Tchekhovskoi, and I. Pletnev. InChI - the worldwide chemical structure identifier standard. *Journal of cheminformatics*, 5(1):7, jan 2013.

[54] C. S. Henry, M. DeJongh, A. A. Best, P. M. Frybarger, B. Linsay, and R. L. Stevens. High-throughput generation, optimization and analysis of genome-scale metabolic models. *Nature Biotechnology*, 28(9):977–82, sep 2010.

[55] C. S. Henry, J. F. Zinner, M. P. Cohoon, and R. L. Stevens. iBsu1103: a new genome-scale metabolic model of Bacillus subtilis based on SEED annotations. *Genome Biology*, 10(6):R69, 2009.

[56] F. Holzschuher and R. Peinl. Performance of graph query languages. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops on - EDBT '13*, page 195, New York, New York, USA, 2013. ACM Press.

[57] K. Hübner, S. Sahle, and U. Kummer. Applications and trends in systems biology in biochemistry. *The FEBS journal*, 278(16):2767–857, aug 2011.

[58] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, a. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, a. a. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley, T. C. Hodgman, J.-H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremling, U. Kummer, N. Le Novere, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, and J. Wang. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, mar 2003.

[59] T. Ideker, T. Galitski, and L. Hood. A New Spproach to Decoding Life: Systems Biology. *Annual Review of Genomics and . . .*, 2001.

[60] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barabási. The large-scale organization of metabolic networks. *Nature*, 407(6804):651–654, oct 2000.

[61] T. Jewison, C. Knox, V. Neveu, Y. Djoumbou, A. C. Guo, J. Lee, P. Liu, R. Mandal, R. Krishnamurthy, I. Sinelnikov, M. Wilson, and D. S. Wishart. YMDB: The yeast metabolome database. *Nucleic Acids Research*, 40(November 2011):815–820, 2012.

[62] N. Juty, N. Le Novère, and C. Laibe. Identifiers.org and MIRIAM Registry: community resources to provide persistent identification. *Nucleic acids research*, 40(Database issue):D580–6, jan 2012.

[63] M. Kanehisa, S. Goto, Y. Sato, M. Kawashima, M. Furumichi, and M. Tanabe. Data, information, knowledge and principle: back to metabolism in KEGG. *Nucleic acids research*, 42(Database issue):D199–205, jan 2014.

[64] P. D. Karp and R. Caspi. A survey of metabolic databases emphasizing the MetaCyc family. *Archives of Toxicology*, 85(9):1015–33, sep 2011.

[65] P. D. Karp, C. A. Ouzounis, C. Moore-Kochlacs, L. Goldovsky, P. Kaipa, D. Ahrén, S. Tsoka, N. Darzentas, V. Kunin, and N. López-Bigas. Expansion of the BioCyc collection of pathway/genome databases to 160 genomes. *Nucleic Acids Research*, 33(19):6083–6089, jan 2005.

[66] P. D. Karp, S. M. Paley, M. Krummenacker, M. Latendresse, J. M. Dale, T. J. Lee, P. Kaipa, F. Gilham, A. Spaulding, L. Popescu, T. Altman, I. T. Paulsen, I. M. Keseler, and R. Caspi. Pathway Tools version 13.0: integrated software for pathway/genome informatics and systems biology. *Briefings in bioinformatics*, 11(1):40–79, jan 2010.

[67] Y. Katsuyama, M. Matsuzawa, N. Funa, and S. Horinouchi. Production of curcuminoids by Escherichia coli carrying an artificial biosynthesis pathway. *Microbiology (Reading, England)*, 154(Pt 9):2620–8, sep 2008.

[68] Z. A. King, J. Lu, A. Dräger, P. Miller, S. Federowicz, J. A. Lerman, A. Ebrahim, B. Ø. Palsson, and N. E. Lewis. BiGG Models: A platform for integrating, standardizing and sharing genome-scale models. *Nucleic Acids Research*, 44(D1):D515–D522, jan 2016.

[69] S. Klamt, U.-U. Haus, and F. Theis. Hypergraphs and cellular networks. *PLoS computational biology*, 5(5):e1000385, may 2009.

[70] V. Kolomičenko, M. Svoboda, and I. H. Mlýnková. Experimental Comparison of Graph Databases. In *Proceedings of International Conference on Information Integration and Web-based Applications & Services - IIWAS '13*, pages 115–124, New York, New York, USA, 2013. ACM Press.

[71] K. T. Konstantinidis and J. M. Tiedje. Trends between gene content and genome size in prokaryotic species with larger genomes. *Proceedings of the National Academy of Sciences*, 101(9):3160–3165, mar 2004.

[72] M. Kotera, M. Hattori, M. Oh, and R. Yamamoto. RPAIR: a reactant-pair database representing chemical changes in enzymatic reactions. *Genome Informatics*, (1):3–4, 2004.

[73] M. Kotera, Y. Okuno, M. Hattori, S. Goto, and M. Kanehisa. Computational assignment of the EC numbers for genomic-scale analysis of enzymatic reactions. *Journal of the American Chemical Society*, 126(50):16487–98, dec 2004.

[74] A. Kumar, P. F. Suthers, and C. D. Maranas. MetRxn : a knowledgebase of metabolites and reactions spanning metabolic models and databases. *BMC Bioinformatics*, 13(1):13, 2012.

[75] C. Laibe and N. Le Novère. MIRIAM Resources: tools to generate and resolve robust cross-references in Systems Biology. *BMC systems biology*, 1:58, jan 2007.

[76] M. Lang, M. Stelzer, and D. Schomburg. BKM-react, an integrated biochemical reaction database. *BMC biochemistry*, 12(1):42, 2011.

[77] M. Latendresse and P. D. Karp. An advanced web query interface for biological databases. *Database: The Journal of Biological Databases and Curation*, 2010:13, jan 2010.

[78] N. Le Novère. Model storage, exchange and integration. *BMC neuroscience*, 7 Suppl 1:S11, jan 2006.

[79] D.-Y. Lee, L. T. Fan, S. Park, S. Y. Lee, S. Shafie, B. Bertók, and F. Friedler. Complementary identification of multiple flux distributions and multiple metabolic pathways. *Metabolic Engineering*, 7(3):182–200, may 2005.

[80] J. A. Lerman, D. R. Hyduke, H. Latif, V. A. Portnoy, N. E. Lewis, J. D. Orth, A. C. Schrimpe-Rutledge, R. D. Smith, J. N. Adkins, K. Zengler, and B. Ø. Palsson. In silico method for modelling metabolism and gene product expression at genome scale. *Nature Communications*, 3(1):929, jan 2012.

[81] D. M. Lowe, P. T. Corbett, P. Murray-Rust, and R. C. Glen. Chemical Name to Structure: OPSIN, an Open Source Solution. *Journal of Chemical Information and Modeling*, 51(3):739–753, mar 2011.

[82] H. Ma and A.-P. Zeng. Reconstruction of metabolic networks from genome data and analysis of their global structure for various organisms. *Bioinformatics (Oxford, England)*, 19(2):270–7, jan 2003.

[83] D. Machado, Z. Soons, K. R. Patil, E. C. Ferreira, and I. Rocha. Random sampling of elementary flux modes in large-scale metabolic networks. *Bioinformatics (Oxford, England)*, 28(18):i515–i521, sep 2012.

[84] S. Magnúsdóttir, A. Heinken, L. Kutt, D. A. Ravcheev, E. Bauer, A. Noronha, K. Greenhalgh, C. Jäger, J. Baginska, P. Wilmes, R. M. T. Fleming, and I. Thiele. Generation of genome-scale metabolic reconstructions for 773 members of the human gut microbiota. *Nature Biotechnology*, 35(1):81–89, nov 2016.

[85] J. W. May, a. G. James, and C. Steinbeck. Metingear: a development environment for annotating genome-scale metabolic models. *Bioinformatics (Oxford, England)*, 29(17):2213–5, sep 2013.

[86] M. H. Medema, R. van Raaphorst, E. Takano, and R. Breitling. Computational tools for the synthetic design of biochemical pathways. *Nature reviews. Microbiology*, 10(3):191–202, mar 2012.

[87] B. Merlet, N. Paulhe, F. Vinson, C. Frainay, M. Chazalviel, N. Poupin, Y. Gloaguen, F. Giacomoni, and F. Jourdan. A Computational Solution to Automatically Map Metabolite Libraries in the Context of Genome Scale Metabolic Networks. *Frontiers in Molecular Biosciences*, 3(February):1–12, feb 2016.

[88] M. L. Mo, B. Ø. Palsson, and M. J. Herrgård. Connecting extracellular metabolomic measurements to intracellular flux states in yeast. *BMC Systems Biology*, 3:37, jan 2009.

[89] J. Monk, J. Nogales, and B. Ø. Palsson. Optimizing genome-scale network reconstructions. *Nature Biotechnology*, 32(5):447–452, may 2014.

[90] S. Moretti, O. Martin, T. Van Du Tran, A. Bridge, A. Morgat, and M. Pagni. MetaNetX/MNXref – reconciliation of metabolites and biochemical reactions to bring together genome-scale metabolic networks. *Nucleic Acids Research*, 44(November 2015):gkv1117, nov 2015.

[91] F. Naumann. Data profiling revisited. *ACM SIGMOD Record*, 42(4):40–49, feb 2014.

[92] N. M. O'Boyle. Towards a Universal SMILES representation - A standard method to generate canonical SMILES based on the InChI. *Journal of cheminformatics*, 4(1):22, jan 2012.

[93] N. M. O'Boyle, M. Banck, C. a. James, C. Morley, T. Vandermeersch, and G. R. Hutchison. Open Babel: An open chemical toolbox. *Journal of cheminformatics*, 3(1):33, jan 2011.

[94] N. M. O'Boyle, R. Guha, E. L. Willighagen, S. E. Adams, J. Alvarsson, J. C. Bradley, I. V. Filippov, R. M. Hanson, M. D. Hanwell, G. R. Hutchison, C. a. James, N. Jeliazkova, A. S. I. D. Lang, K. M. Langner, D. C. Lonie, D. M. Lowe, J. Pansanel, D. Pavlov, O. Spjuth, C. Steinbeck, A. L. Tenderholt, K. J. Theisen, and P. Murray-Rust. Open Data, Open Source and Open Standards in chemistry: The Blue Obelisk five years on. *Journal of Cheminformatics*, 3(1):37, 2011.

[95] E. J. O'Brien, J. M. Monk, and B. Ø. Palsson. Using Genome-scale Models to Predict Biological Capabilities. *Cell*, 161(5):971–987, may 2015.

[96] B. G. Olivier, J. M. Rohwer, and J.-H. S. Hofmeyr. Modelling cellular systems with PySCeS. *Bioinformatics*, 21(4):560–561, feb 2005.

[97] J. D. Orth, T. M. Conrad, J. Na, J. A. Lerman, H. Nam, A. M. Feist, and B. Ø. Palsson. A comprehensive genome-scale reconstruction of Escherichia coli metabolism–2011. *Molecular systems biology*, 7(535):535, jan 2011.

[98] J. D. Orth, I. Thiele, and B. Ø. Palsson. What is flux balance analysis? *Nature biotechnology*, 28(3):245–8, mar 2010.

[99] R. A. Overbeek, T. Begley, R. M. Butler, J. V. Choudhuri, H.-Y. Chuang, M. Cohoon, V. de Crécy-Lagard, N. Diaz, T. Disz, R. A. Edwards, M. Fonstein, E. D. Frank, S. Gerdes, E. M. Glass, A. Goesmann, A. Hanson, D. Iwata-Reuyl, R. Jensen, N. Jamshidi, L. Krause, M. Kubal, N. Larsen, B. Linke, A. C. McHardy, F. Meyer, H. Neuweger, G. J. Olsen, R. Olson, A. L. Osterman, V. Portnoy, G. D. Pusch, D. A. Rodionov, C. Rückert, J. Steiner, R. L. Stevens, I. Thiele, O. Vassieva, Y. Ye, O. Zagnitko, and V. Vonstein. The subsystems approach to genome annotation and its use in the project to annotate 1000 genomes. *Nucleic Acids Research*, 33(17):5691–702, jan 2005.

[100] S. Pabinger, R. Snajder, T. Hardiman, M. Willi, A. Dander, and Z. Trajanoski. MEMOSys 2.0: an update of the bioinformatics database for genome-scale models and genomic data. *Database*, 2014(February):bau004–bau004, feb 2014.

[101] P. Pharkya, A. P. Burgard, and C. D. Maranas. OptStrain : A computational framework for redesign of microbial production systems. *Genome Research*, (814):2367–2376, 2004.

[102] J. Pinho, J. L. Sobral, and M. Rocha. Parallel evolutionary computation in bioinformatics applications. *Computer methods and programs in biomedicine*, 110(2):183–91, may 2013.

[103] E. Pitkänen, P. Jouhten, and J. Rousu. Inferring branching pathways in genome-scale metabolic networks. *BMC systems biology*, 3:103, jan 2009.

[104] K. D. Pruitt, G. R. Brown, S. M. Hiatt, F. Thibaud-Nissen, A. Astashyn, O. Ermolaeva, C. M. Farrell, J. Hart, M. J. Landrum, K. M. McGarvey, M. R. Murphy, N. a. O'Leary, S. Pujar, B. Rajput, S. H. Rangwala, L. D. Riddick, A. Shkeda, H. Sun, P. Tamez, R. E. Tully, C. Wallin, D. Webb, J. Weber, W. Wu, M. DiCuccio, P. Kitts, D. R. Maglott, T. D. Murphy, and J. M. Ostell. RefSeq: an update on mammalian reference sequences. *Nucleic acids research*, 42(Database issue):D756–63, jan 2014.

[105] A. Ravikrishnan and K. Raman. Critical assessment of genome-scale metabolic networks: the need for a unified standard. *Briefings in Bioinformatics*, 16(6):1057–1068, nov 2015.

[106] J. L. Reed, T. R. Patel, K. H. Chen, A. R. Joyce, M. K. Applebee, C. D. Herring, O. T. Bui, E. M. Knight, S. S. Fong, and B. Ø. Palsson. Systems approach to refining genome annotation. *Proceedings of the National Academy of Sciences of the United States of America*, 103(46):17480–4, nov 2006.

[107] I. Rocha, P. Maia, P. Evangelista, P. Vilaça, S. Soares, J. P. Pinto, J. Nielsen, K. R. Patil, E. C. Ferreira, and M. Rocha. OptFlux: an open-source software platform for in silico metabolic engineering. *BMC systems biology*, 4:45, jan 2010.

[108] G. Rodrigo, J. Carrera, K. L. J. Prather, and A. Jaramillo. DESHARKY: automatic design of metabolic pathways for optimal cell growth. *Bioinformatics (Oxford, England)*, 24(21):2554–6, nov 2008.

[109] N. Rodriguez, A. Thomas, L. Watanabe, I. Y. Vazirabad, V. Kofia, H. F. Gómez, F. Mittag, J. Matthes, J. Rudolph, F. Wrzodek, E. Netz, A. Diamantikos, J. Eichner,

R. Keller, C. Wrzodek, S. Fröhlich, N. E. Lewis, C. J. Myers, N. Le Novère, B. Ø. Palsson, M. Hucka, and A. Dräger. JSBML 1.0: providing a smorgasbord of options to encode systems biology models. *Bioinformatics*, 31(20):3383–3386, oct 2015.

[110] J. Schellenberger, J. O. Park, T. M. Conrad, and B. Ø. Palsson. BiGG: a Biochemical Genetic and Genomic knowledgebase of large scale metabolic reconstructions. *BMC Bioinformatics*, 11:213, jan 2010.

[111] J. Schellenberger, R. Que, R. M. T. Fleming, I. Thiele, J. D. Orth, A. M. Feist, D. C. Zielinski, A. Bordbar, N. E. Lewis, S. Rahmanian, J. Kang, D. R. Hyduke, and B. Ø. Palsson. Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox v2.0. *Nature protocols*, 6(9):1290–307, sep 2011.

[112] C. H. Schilling, J. S. Edwards, and B. Ø. Palsson. Toward metabolic phenomics: analysis of genomic data using flux balances. *Biotechnology progress*, 15(3):288–95, 1999.

[113] C. H. SCHILLING and B. Ø. PALSSON. Assessment of the Metabolic Capabilities of Haemophilus influenzae Rd through a Genome-scale Pathway Analysis. *Journal of Theoretical Biology*, 203(3):249–283, apr 2000.

[114] S. Schuster and C. Hlgetag. On Elementary Flux Modes in Biochemical Reaction Systems at Steady State. *Journal of Biological Systems*, 2(2):165–182, 1994.

[115] D. Segrè, J. Zucker, J. Katz, X. Lin, P. D'Haeseleer, W. P. Rindone, P. Kharchenko, D. H. Nguyen, M. A. Wright, and G. M. Church. From Annotated Genomes to Metabolic Flux Models and Kinetic Parameter Fitting. *OMICS: A Journal of Integrative Biology*, 7(3):301–316, sep 2003.

[116] K. Smallbone, E. Simeonidis, D. S. Broomhead, and D. B. Kell. Something from nothing: bridging the gap between constraint-based and kinetic modelling. *The FEBS journal*, 274(21):5576–85, nov 2007.

[117] C. Steinbeck, Y. Han, S. Kuhn, O. Horlacher, E. Luttmann, and E. Willighagen. The Chemistry Development Kit (CDK): an open-source Java library for Chemo- and Bioinformatics. *Journal of chemical information and computer sciences*, 43(2):493–500, 2003.

[118] M. Sud, E. Fahy, D. Cotter, A. Brown, E. A. Dennis, C. K. Glass, A. H. Merrill, R. C. Murphy, C. R. H. Raetz, D. W. Russell, and S. Subramaniam. LMSD: LIPID MAPS structure database. *Nucleic Acids Research*, 35(Database):D527–D532, jan 2007.

[119] N. Swainston, R. Batista-Navarro, P. Carbonell, P. D. Dobson, M. Dunstan, A. J. Jervis, M. Vinaixa, A. R. Williams, S. Ananiadou, J.-L. Faulon, P. Mendes, D. B. Kell, N. S. Scrutton, and R. Breitling. biochem4j: Integrated and extensible biochemical knowledge through graph databases. *PLOS ONE*, 12(7):e0179130, jul 2017.

[120] M. Terzer and J. Stelling. Large-scale computation of elementary flux modes with bit pattern trees. *Bioinformatics (Oxford, England)*, 24(19):2229–35, oct 2008.

[121] T. W. J. M. van Herpen, K. Cankar, M. Nogueira, D. Bosch, H. J. Bouwmeester, and J. Beekwilder. Nicotiana benthamiana as a production platform for artemisinin precursors. *PloS one*, 5(12):e14222, jan 2010.

[122] W. A. Warr. Representation of chemical structures. *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 1(4):557–579, jul 2011.

[123] D. Weininger. SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *Journal of Chemical Information and Modeling*, 28(1):31–36, feb 1988.

[124] D. S. Wishart, T. Jewison, A. C. Guo, M. Wilson, C. Knox, Y. Liu, Y. Djoumbou, R. Mandal, F. Aziat, E. Dong, S. Bouatra, I. Sinelnikov, D. Arndt, J. Xia, P. Liu, F. Yallou, T. Bjorndahl, R. Perez-Pineiro, R. Eisner, F. Allen, V. Neveu, R. Greiner,

and A. Scalbert. HMDB 3.0–The Human Metabolome Database in 2013. *Nucleic Acids Research*, 41(D1):D801–D807, jan 2013.

[125] G. Wohlgemuth, P. K. Haldiya, E. Willighagen, T. Kind, and O. Fiehn. The Chemical Translation Service–a web-based tool to improve standardization of metabolomic reports. *Bioinformatics (Oxford, England)*, 26(20):2647–8, oct 2010.

[126] L. Wu, J. Perkins, and G. Schyns. Alternative butanol production process in a microbial cell, 2010.

[127] J. C. Xavier, K. R. Patil, and I. Rocha. Integration of Biomass Formulations of Genome-Scale Metabolic Models with Experimental Data Reveals Universally Essential Cofactors in Prokaryotes. *Metabolic Engineering*, 39(October 2016):200–208, jan 2017.