From Building Blocks to Molecules

A Computational Framework for Modeling Pharmaceutically Relevant Chemical Space

Dissertation
with the aim of achieving the degree

Dr. rer. nat.

at the Faculty of Mathematics, Computer Science and Natural Sciences
Department of Informatics
of Universität Hamburg

submitted by
Florian Lauck
born in Quierschied

Hamburg, July 2016

Gutachter

Prof. Dr. Matthias Rarey
Prof. Dr. Norbert Ritter

Tag der Disputation

25. November 2016

Eidesstattliche Versicherung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Dissertationsschrift selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Hamburg, den 26. Juli 2016

Florian Lauck

Abstract

In drug development, there is a constant demand for new, synthesizable molecules with desirable properties. Molecules with a certain physicochemical profile have a higher probability of having a biological effect, i.e., the ability to enter the body and to bind to a specific protein. Computational tools present a convenient and efficient way to quickly generate such molecules. In this thesis, two new deterministic algorithms were developed to facilitate the generation of large libraries of promising, synthesizable molecules. They both utilize the fragment space model, which describes a chemical space by a set of fragments and rules. The latter determine how fragments can be connected. Due to their combinatorial nature, fragment spaces may represent virtually infinite chemical space.

The first algorithm takes a fragment space as input and exhaustively enumerates all molecules with a user-defined physicochemical profile. It is implemented with constant main memory requirements by utilizing file-based data structures in order to enable the enumeration of millions of molecules. For this purpose, several enumeration experiments were carried out using various physicochemical constraints. The second algorithm constructs fragment spaces from molecular building blocks and synthetic chemical reactions. Molecules retrieved from such spaces have a high likelihood of being synthesizable because they were assembled based on actual chemical reactions. In addition, this information can be translated into a synthetic route for each molecule. Several fragment spaces were constructed from well-established synthetic reactions.

Finally, the third topic of this thesis is the development of a graphical user interface that combines several fragment space methods. This tool, called Fragment Space Commander, enables users to create, visualize and edit fragment spaces and generate new molecules in a convenient way.

Kurzfassung

In der Arzneimittelforschung besteht ständiger Bedarf an neuen, synthetisierbaren Molekülen mit erstrebenswerten Eigenschaften. Moleküle mit bestimmten physikochemischen Merkmalen besitzen mit höherer Wahrscheinlichkeit eine biologische Wirkung, das heißt, die Fähigkeit vom Körper aufgenommen zu werden und sich an ein bestimmtes Protein zu binden. Computergestützte Verfahren stellen hierbei eine einfache und effiziente Möglichkeit dar, solche Moleküle schnell zu generieren. In der vorliegenden Dissertation wurden zwei neue deterministische Algorithmen entwickelt, um die Generierung großer Bibliotheken von erfolgsversprechenden, synthetisierbaren Molekülen zu ermöglichen. Beide Algorithmen nutzen das Fragmentraum-Modell, welches einen chemischen Raum anhand einer Menge von Fragmenten und Regeln beschreibt. Diese Regeln wiederum definieren, wie Fragmente miteinander verbunden werden können. Aufgrund ihres kombinatorischen Charakters können Fragmenträume einen praktisch unbegrenzten chemischen Raum darstellen.

Der erste Algorithmus arbeitet mit einem Fragmentraum als Eingabe und enumeriert vollständig alle Moleküle mit benutzerdefinierten physikochemischen Eigenschaften. Um die Enumeration von Millionen von Molekülen zu ermöglichen, wurde dieser Algorithmus mit konstantem Arbeitsspeicherbedarf unter Verwendung von dateibasierten Datenstrukturen implementiert. Zu diesem Zweck wurden mehrere Experimente mit einer Reihe von verschiedenen physikochemischen Randbedingungen durchgeführt. Der zweite Algorithmus erstellt Fragmenträume aus molekularen Bausteinen und Synthesereaktionen. Die aus solchen Räumen gewonnenen Moleküle weisen eine hohe Wahrscheinlichkeit auf, im Labor herstellbar zu sein. Dies liegt daran, dass sie basierend auf Information zusammengebaut wurden, die von tatsächlichen chemischen Reaktionen abgeleitet wurde. Des Weiteren kann diese Information genutzt werden, um für jedes Molekül einen Syntheseweg zu entwickeln. Mit dieser Methode wurden mehrere Fragmenträume aus gängigen Reaktionen erstellt.

Gegenstand der vorliegenden Dissertation ist schließlich auch die Entwicklung einer grafischen Benutzeroberfläche, welche verschiedene Fragmentraum-Methoden kombiniert. Dieses Tool, genannt Fragment Space Commander, ermöglicht es Nutzern, Fragmenträume zu erstellen, zu visualisieren und zu bearbeiten und, darüber hinaus, neue Moleküle auf einfache Art und Weise zu generieren.

Publications

Articles in Scientific Journals

Florian Lauck and Matthias Rarey. “FSees: Customized Enumeration of Chemical Subspaces with Limited Main Memory Consumption”. In: *Journal of Chemical Information and Modeling* 56.9 (Sept. 2016), pp. 1641–1653

Book Chapters

Florian Lauck and Matthias Rarey. “Coping with Combinatorial Space in Molecular Design”. In: *De novo Molecular Design*. Weinheim, Germany: Wiley-VCH Verlag GmbH & Co. KGaA, Oct. 2013, pp. 325–347

Talks

Florian Lauck. “Exploring local chemical space: From fragments to molecules with a defined physicochemical profile”. In: *Fragment-based Lead Discovery Conference*. Basel, Switzerland, Sept. 2014

Posters

Florian Lauck and Matthias Rarey. “Dealing with Combinatorial Chemical Space: Towards a Universal Framework”. In: *6th Joint Sheffield Conference on Chemoinformatics*. Sheffield, United Kingdom, July 2013

Florian Lauck and Matthias Rarey. “Design Your Chemical Universe: Mining Fragment Spaces for Novel Molecules”. In: *20th European Symposium on Quantitative Structure-Activity Relationship*. St. Petersburg, Russia, Sept. 2014

Danksagung

Ich danke Matthias Rarey für die Möglichkeit dieses interessante Thema zu bearbeiten und für die fachlich exzellente Betreuung. Ausserdem dafür, dass er für Diskussionen und Rat jederzeit zur Verfügung stand und für das angenehme Arbeitsklima in seiner Gruppe. Ich danke auch meinen ehemaligen Kollegen für die sehr angenehme Arbeitsatmosphäre, die zahlreichen fachlichen Diskussionen, aber auch die vielen lustige Momente, an die ich mich gerne zurückerinnern werde. Ich danke allen, die an der Entwicklung der NAOMI Bibliothek vor und während meiner Promotion beteiligt waren und ohne deren Beiträge diese Arbeit nicht möglich gewesen wäre, insbesondere Christian, Stefan, Matthias, Therese, Thomas, Kai und Sascha. Ich danke Nicholas und Oliv für die Rechnungen mit Pipeline Pilot. Ich danke Janna, Melanie und Sarah für alles Organisatorische und Christian für den technischen Support. Ich danke meinen Korrekturlesern Eva, Janna, Kai, Karen und Robert, ohne die die vorliegende Arbeit in dieser Form nicht möglich gewesen wäre. Ich danke Nadine und Jens, ohne die ich nicht in Hamburg gelandet wäre. Ich danke allen, die meine wissenschaftlichen Karriere beeinflusst haben, meinen Lehrern und Betreuern aus Schule und Studium, meinen ehemaligen Chefs und Kollegen der UCSF, und meinem Onkel Rudi.

Ich möchte an dieser Stelle auch meinen Freunden und meiner Familie danken, insbesondere meiner Mutter Mariele und meinen Brüdern Felix und Maximilian. Danke für die Unterstützung und den Rückhalt den ihr mir gegeben habt. Als letztes möchte ich Janna danken, die mich durch die gesamte Promotion begleitet hat. Danke für deine Unterstützung und Geduld, vor allem in der letzten Phase.

Contents

Abbreviations

ADME(T)	absorption, distribution, metabolism, excretion, (toxicology)
API	application programming interface
ATP	adenosine triphosphate
AUC	area under the curve
BLOB	binary large object
CAS	Chemical Abstract Service
CPU	central processing unit
DFS	depth-first search
DNA	deoxyribonucleic acid
ECFP	extended-connectivity fingerprint
FBLD	fragment-based lead design
FCFP	functional class fingerprint
FSDB	fragment space database
FSees	Fragment Space exhaustive enumeration system
FSF	fragment space format
GB	GigaByte
GUI	graphical user interface
HELLS	Hamburg enumerated lead-like set
HTS	high-throughput screening
KDS	known drug space
MB	MegaByte
MW	molecular weight (more precisely: molecular mass)
NMR	nuclear magnetic resonance
PDB	Protein Data Bank
PMI	principle moment of inertia
QSAR	quntitative structure activity relationship
RAM	random access memory
RNA	ribonucleic acid
Ro3	Rule of three
Ro5	(Lipinski's) Rule of five
ROC	receiver operating charactersitic
SMARTS	SMILES arbitrary target specification
SMILES	simplified molecular-input line-entry system
TT	topological torsions
VHTS	virtual high-throughput screening

1 Introduction

1.1 Motivation

The search for medicinal drugs has changed dramatically over the past century. It has evolved from a purely empirical process of trying different herbal and animal products into a highly complex, interdisciplinary endeavor [6]. The goal of rational drug design is to develop small organic molecules that bind to a specific protein and exhibit ideal properties regarding bioavailability. For the most part, this process is based on well-established experimental techniques. In recent decades more and more computational tools were incorporated in the drug development workflow because they allow researchers to much more quickly assess a situation. These tools help to enhance the drug development process as idea-generating tools, especially in early stages of the process [7, 8]. At the beginning, knowledge about function-determining features of desired ligand molecules is scarce and new ideas are required that lead to patentable molecules. In this phase, similarity searching, pharmacophore mapping, or virtual screening are usually applied (see 2.2). These techniques are intended to identify initial hits and require virtual compound libraries as input.

Due to the enormous extend of the chemical space, it is not possible to simply synthesize or computationally generate all molecules and test them. Bohacek *et al.* estimate the size of the interesting chemical space (i.e. molecules with up to 30 atoms) with 10^{60} molecules [9]. The "virtual chemistry space" is estimated to even contain 10^{100} molecules [10]. On the other hand, the "known drug space", i.e., molecules that may actually be an active substance with a desirable effect, is estimated with *only* $1.1 - 2.0 * 10^6$ molecules [11]. Trying to generate all molecules in order to find the desired ones by brute-force computation quickly leads to combinatorial explosion. To compile a virtual molecular library for further testing, only molecules with desirable properties must be considered to increase the chance that the resulting set of molecules represents a promising chemical subspace. The common strategy is to incorporate certain constraints in the building process in order to limit the search space.

Depending on the progress of the campaign and how much is known about the target protein, constraints can be derived from different sources. In the context of library design, physicochemical properties of molecules play an important role [12–19]. They are ideal constraints since they are single numeric values that can be measured experimentally for actual molecules and computed efficiently for virtual ones. In addition, certain properties serve as indicators for bioavailability [13, 15–18], i.e., the ability of a molecule to enter into an organism and arrive at the target location.

In addition to physicochemical properties, the ability to synthesize a molecule is of utmost importance. A predicted molecule may exhibit the most ideal properties in theory but if it cannot be created in the laboratory it is worthless. Unfortunately, this is neglected by many computational methods. By including information about chemical reactions, an increased chance of synthetic tractability is achieved [20, 21].

In the laboratory, reactions are used to create larger molecules from smaller ones. These small organic molecules are called fragments and can be incorporated in the design process from the beginning. In recent years, fragment-based methods have gained popularity in drug discovery and have been successfully applied [7, 22, 23]. Fragments are an interesting basis for experimental and computational drug development for several reasons: They are accessible; vendors offer large libraries of so called building blocks, i.e., small molecules for the use in synthetic reactions. They are chemically motivated constraints that limit the search space at no extra (computational) cost. A fragment describes a set of atoms that form a chemically stable, functional unit. These units can be combined into larger entities.

In this work, fragments are utilized to construct novel molecules. The computational methodology for working with them is a *fragment space*. A fragment space is an implementation of a combinatorial space consisting of molecular fragments and connection rules derived from reactions or retrosynthetic rules. By storing only the building blocks and connection rules, a fragment space represents a potentially very large amount of molecules in a very space efficient manner. In order to retrieve the interesting molecules from such a space, algorithms are required that implement strategies to avoid combinatorial explosion. A number of such algorithms for very different use cases already exist They include similarity searching [24], structure based molecular design [25, 26], scaffold replacement [27, 28], enumeration [29, 30], and substructure searching [31–33]. For the construction of fragment spaces, an automated method for retrosynthetic fragmentation exists [34]. Synthetic reaction information has been used solely to manually construct a fragment space [35]. Figure 1.1 gives an overview of the existing methods and the ones that are subject to this thesis.

Apart from designing promising molecules, it is very important to make computational tools available to researchers for convenient access. Chemists, in particular, may have reservations regarding computational methods. Providing tools with a command-line interface enables the use of pipelining tools, but discourages users without computational background from using – or even trying – them. In order to raise acceptance for cheminformatics methods, their usage should not only be possible for cheminformaticians but also for users without computational training. This requires user-friendly interfaces with sensible default parameters. Graphical user interfaces (GUIs) constitute a much more convenient and contemporary way of accessing scientific algorithms. Especially for the researchers in the laboratory, they provide additional value due to visualization of both input and output data for instant inspection.

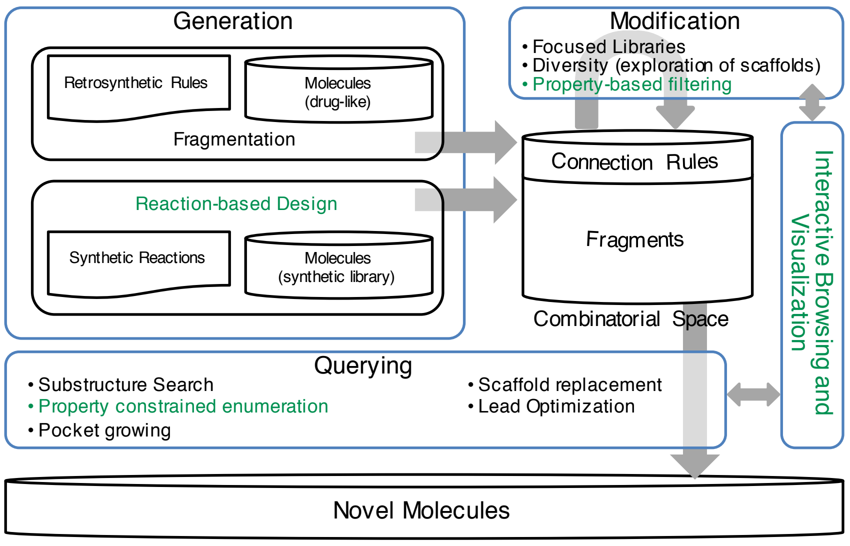


Figure 1.1: Overview of the combinatorial space pipeline leading to novel molecules. Rounded blue boxes hold strategies and applications for generation, manipulation, and querying. Arrows denote the flow of information. The method highlighted in green are the subject of this thesis. *This figure is an updated version of a figure originally published in [2].*

1.2 Research Goals

In drug development, there is a constant need for new ideas for marketable drugs. The main challenge is to design novel (i.e. not patented), target specific, bioactive, and chemically feasible molecules. The problem addressed in this thesis is the generation of large amounts of molecules for the use as input to (high-throughput) methods that are designed to identify interesting candidates in the early stages of the drug development pipeline. Due to the size of chemical space, a brute force approach of generating all possible molecules is not feasible. Instead, constraints must be applied in order to limit the search space to interesting candidates. In addition, reaction based information must be incorporated to increase the likelihood of a molecule to be synthesizable. This should be realized in the context of fragment spaces because this methodology allows to separate the generation of the combinatorial chemical space from the method to generate molecules. A valid fragment space – no matter how it was generated – can be the input to any of the existing algorithms [24, 25, 27, 29, 31, 32].

Three goals are defined:

1. Development of a new, deterministic algorithm to enumerate all possible molecules that are represented by a fragment space and a set of ranges of physicochemical constraints such as molecular weight, number of hydrogen bonding groups, or calculated logP. The algorithm must be able to enumerate large chemical spaces and create large amounts of molecules.

2. Development of an automated method for generating combinatorial space in the form of fragment spaces based on information about synthetic chemical reactions. By utilizing reaction information, the chemical feasibility of molecules retrieved from such a fragment space – regardless of the query method – is increased.

3. Development of a graphical user interface for working with fragment spaces. The tool should provide functionality to create, visualize, and invoke new and existing methods for querying fragment spaces.

1.3 Outline

This thesis is structured as follows. Chapter 2 introduces the basic principles of drug discovery, including the biochemical basics relevant for understanding the underlying molecular processes, the typical drug discovery pipeline as employed by many pharmaceutical companies, and the principles of fragment-based lead discovery (FBLD). In chapter 3 the concept of chemical and combinatorial space is introduced including a discussion of the number of interesting and possible molecules. Furthermore, fragment space as a way of modeling combinatorial space is introduced. Then, existing strategies and algorithms for exploring chemical space are discussed in chapter 4. Chapter 5 follows with a description of how molecules and molecular properties are modeled in order to describe chemical space. Next, the newly developed algorithms for reaction-based construction and constraint-based enumeration of fragment spaces as well as the re-implemented algorithm for retrosynthetic fragmentation are described in chapter 6. Chapter 7 introduces the different sources of data used for the validation and experiments discussed in chapter 8. The latter discusses several experiments utilizing the previously developed algorithms and describes the optimization process of the enumeration. In chapter 9 the graphical user interface for working with fragment spaces, Fragment Space Commander, is introduced. Furthermore, all command-line tools that implement either an algorithm or functionality used for the analysis of data are described. Finally, chapter 10 summarizes the work and discusses possible improvements.

2 Principles of Drug Discovery

Drug discovery is a multidisciplinary endeavor where the fields of biology, chemistry, medicine, and more recently informatics, in the form of bio- and cheminformatics, are joined. The development of new drug molecules requires a detailed understanding of biochemical processes that can lead to disease and the ability to design organic molecules that can interact with macromolecules, such as proteins. Informatics provides computational models of an organism – or parts thereof – to speed up the development process by shifting experiments from the laboratory into the computer. On what information a model is based on and what it represents depends on the specific use case. This chapter describes the traditional drug discovery pipeline and will elucidate how computational methods can provide assistance in this process. Then, the features of fragment-base lead discovery are highlighted. First, an introduction to the most important biochemical principles and terminology is provided.

2.1 Biochemical Background

Every living organism is composed of a multitude of molecules. They range from very small inorganic molecules such as water or ions, over medium-sized, organic molecules such as sugars or lipids, to macromolecules such as proteins, ribonucleic acid (RNA), and deoxyribonucleic acid (DNA). Proteins often congregate to even larger functional units, e.g., anti bodies, the nuclear pore complex, or virus capsids [36–38].

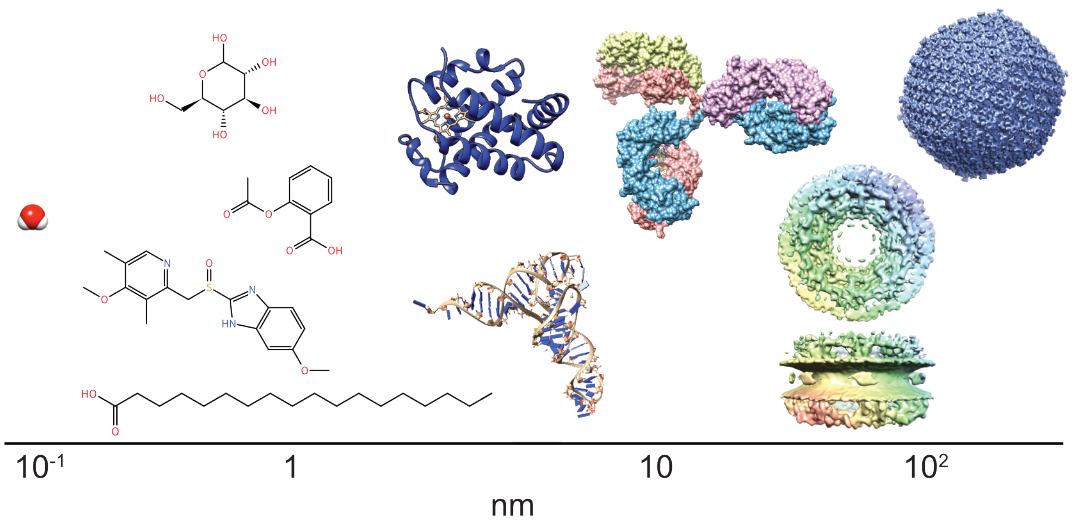


Figure 2.1: Water, Glucose, acetylsalicylic acid (Aspirin), Omeprazole, lipid, hemoglobin, RNA, antibody, nuclear pore complex, and Faustovirus capsid.

While DNA is a passive molecule, proteins – and some types of RNA – are actively involved in the processes of an organism. Therefore, they are the major target in drug development efforts. They are responsible for a vast array of cellular functions including structure and motility, catalysis of chemical reactions (enzymes), transport and storage of molecules, regulation of cellular processes, the immune system, cell-cell recognition, signaling and DNA replication. They consist of one or more chains of amino acid residues that are connected via peptide bonds. These bonds are formed when the amine and the carboxylic acid functional groups of two residues react (see Figures 6.3 and 6.4). Proteins consist of 20 different naturally occurring amino acids, which are connected according to the order encoded in the DNA. The sequence of amino acid residues determines the 3D shape into which the chain folds. This 3D shape, i.e., the *structure* of a protein, determines its function.

Small organic molecules also fulfill a variety of functions in an organism. They serve as building blocks of proteins, RNA, and DNA (amino acids and nucleotides), as carriers of energy (e.g. ATP), as energy storage (e.g. triglycerides), as biochemical messengers (e.g. hormones), and others. Small molecules can either be produced by the organism itself or absorbed from the environment, for example by ingesting food. In either case, proteins are involved. They interact with these molecules in order to detect, transport, and metabolize them.

A small molecule that binds to a protein is called a *ligand*. Ligand binding can be reversible or irreversible. For an irreversible interaction, the ligand and the protein chemically react to form a covalent bond. In the reversible case, a molecule and a protein form a *complex*, meaning that both molecules remain separate chemical entities. Instead of a covalent bond, the interaction comprises of attractive and repulsive non-covalent intermolecular forces, i.e., hydrogen bonds, ionic interactions, metal complexes, hydrophobic interactions, and cation-π interactions. Complementary interaction profiles in a protein and a ligand lead to the formation of a protein-ligand-complex, for which the overall energy is lower than for both molecules in solution. This is often compared to a *lock*, the protein, and a *key*, the ligand. Emil Fischer first coined the term lock-and-key principle in 1894 [38]. The goal of rational drug design is to develop new molecules with complementary features so that they bind to proteins involved in a disease. Depending on the disease and the proteins involved, a new molecule can enable or prevent the function of a protein. The major challenge is to design a molecule that is specific to the protein in question. If the molecule binds to other proteins as well, this most likely leads to undesired *side effects*.

For certain protein classes, ligands are named after their function, i.e., the effect on the protein. In the context of enzymes – the largest class of drug targets [39] – a ligand is either a *substrate* or an *inhibitor*. Substrates are molecules for which the enzyme catalyzes a reaction. An inhibitor, on the other hand, binds to the enzyme and prevents, i.e., *inhibits*, its catalytic function. The mechanism of action can be different. A *competitive inhibitor* binds to the same site as a substrate while blocking it. *Uncompetitive* and *non-competitive inhibitors* bind at other sites of the enzyme. The former molecule binds to both the enzyme and the enzyme-substrate-complex, while the latter binds exclusively to the complex. A drug molecule can be of either type [39]. The inhibition is initiated either by the fact that a molecule blocks a site that is required for the proper function, or by *allosteric inhibition*, i.e., by introducing a conformational change in the protein and therefore changing its structure. The second largest class of drug targets are receptors, i.e., proteins that receive chemical signals from outside a cell. Ligands that bind to a receptor and trigger a response within the cell are called *agonists*. Their counterpart are *antagonists*, i.e., ligands that bind, but prevent the function of the receptor. Another type of ligand is an *inverse agonist*. It induces a response opposite to that of the agonist. This is only possible in case the receptor has a constitutive activity, i.e., a basic activity in the absence of any agonist. For this class too, drug molecules can be of either type [39].

2.2 The Drug Discovery Pipeline

The drug discovery pipeline can be divided into several phases in which different experimental and computational methods are applied. In each phase, the number of potential new drug candidates is reduced. In the following, the different phases are described, a depiction is shown in Figure 2.2.

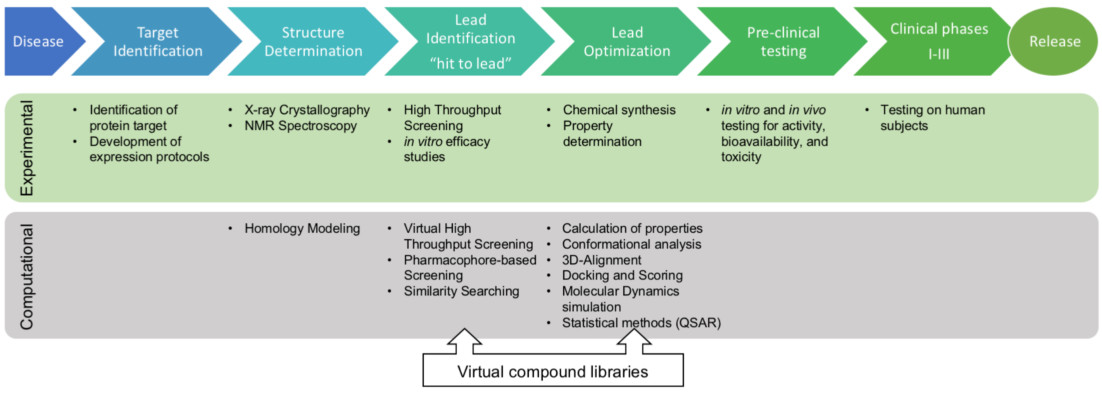

Disease
Target Identification
Structure Determination
Lead Identification "hit to lead"
Lead Optimization
Pre-clinical testing
Clinical phases I-III
Release
Experimental
Identification of protein target
Development of expression protocols
X-ray Crystallography
NMR Spectroscopy
High Throughput Screening
in vitro efficacy studies
Chemical synthesis
Property determination
in vitro and in vivo testing for activity, bioavailability, and toxicity
Testing on human subjects
Computational
Homology Modeling
Virtual High Throughput Screening
Pharmacophore-based Screening
Similarity Searching
Calculation of properties
Conformational analysis
3D-Alignment
Docking and Scoring
Molecular Dynamics simulation
Statistical methods (QSAR)
Virtual compound libraries

Figure 2.2: Generic Drug Discovery Pipeline with experimental and computational methods listed. Virtual compound libraries as generated by the enumeration method in this thesis represent the input for many of the computational methods listed.

2.2.1 Target Identification

The first step towards developing a therapeutic molecule is to investigate the disease on a molecular level. This investigation is mostly done in the laboratory by applying experimental techniques from molecular biology, biochemistry and physics (i.e. imaging). The knowledge gained in this phase establishes the basis for the rational drug design process. First, one must identify proteins or a protein pathway involved in the disease in order to find a protein that can be influenced. Depending on the disease, this can be a human protein, for example in the case of cancer, or a protein of a pathogen such as a bacterium or a virus. Then, a target must be chosen from this set having the right properties. Since the goal is to develop a ligand that can bind to the protein in order to affect its activity and/or conformation, usually proteins are chosen that already interact with small molecules and therefore exhibit distinct binding sites. Such proteins are enzymes, receptors, or transporters [39, 40]. It is important that this binding site exhibits a certain specificity so that a molecule can be designed that binds exclusively to this protein. If the protein natively binds different molecules – if it is *promiscuous* – its binding site will be less specific and harder to target. Similarly, if the designed ligand binds to multiple proteins, it may cause side effects in the patient. Neither case is desirable. Furthermore, the protein should have a very slow mutation rate, so that it does not change during the lengthy drug development process. This is especially critical when targeting viral or bacterial diseases. Finally, it should be possible to express and isolate the protein in large amounts so that it is available in later stages of the pipeline, e.g., for the production of protein crystals, high throughput screening, or affinity measurements during lead optimization.

2.2.2 Structure Determination

The next step is the determination of the 3D structure of the selected protein. It is the basis for structure-based methods since it allows to examine the function of a protein and the exploitable structural features. Known structures can be found in the publicly available Protein Data Bank[1] [41] (PDB) or in an in-house database. If no structure is available, it can be determined via experimental techniques such as X-ray crystallography or nuclear magnetic resonance (NMR) spectroscopy. The former is the most widely used technique and requires that the protein has been crystallized. If crystallization is not possible, NMR spectroscopy can be used. It is more elaborate since it requires isotopic labeling of the molecule. In addition, it is only applicable to relatively small macromolecules. In the case that the structure cannot be determined by experiment, homology modeling can be used. This is a computational method to generate a structure from the sequence of the protein and a known homologous structure, e.g., a protein with the same function from different organisms. In general, homologous proteins are related because of a shared ancestry. If a structure can neither be determined nor generated, only ligand-based methods can be applied in the next steps.

2.2.3 Lead Identification

The goal in this phase is to find molecules that can be used as starting points for the development of potent binders. For this purpose, the exact mechanism of action for known ligands is investigated first. Then, an array of experimental and computational methods is used to find non-naturally occurring and synthesizable molecules. Most prominently, this is high throughput screening (HTS) and virtual high throughput screening (VHTS). HTS is an experimental method in which each molecule of a compound library is tested for activity against the protein under investigation. This is a mostly automated process, which requires large amounts of purified protein. Virtual high throughput screening (short: virtual screening) is the computational equivalent to HTS. It uses a model of the 3D structure of a protein and a virtual library of molecules. The individual molecules are placed into the binding pocket by a docking algorithm. Then, the binding mode is scored by means of a scoring function that assesses the strength of the non-covalent interactions between the molecule and the protein. The output of both screening methods is a list with measured or computed binding affinities,

[1]http://www.rcsb.org

respectively. Virtual screening is usually applied first since it is cheaper and faster. In case a protein structure is not available, pharmacophore based methods can be applied. This is an abstract, spatial description of molecular features that form interactions with the target protein and are thus relevant for a pharmacological effect. Such a model can be built from known ligands or homologous proteins. In order to compile a virtual compound library after an initial hit was found, a number of computational techniques to generate novel patentable molecules can be applied (see chapter 4).

The techniques described in this chapter are usually not applied once and in a specific order, but rather in an incremental and interlinked fashion to reduce the number of potential molecules to a manageable amount [42]. They act as filters and idea generation tools for the expert. Ultimately, it takes the knowledge and experience of a medicinal chemist to select molecules that can be further modified and optimized.

2.2.4 Lead Optimization

During the optimization phase, a lead molecule is developed into a drug molecule with the necessary properties to go into clinical trial. The goal is to increase affinity and specificity towards the target protein and to improve absorption and duration of action in the organism. At the same time, side effects and toxicity must be reduced, if not eliminated. While a variety of structural modifications can be applied, it is important to keep the interaction profile unchanged or only slightly changed. This is accomplished by exchanging parts of the molecule with bioisosteres, i.e., groups with similar steric and electronic properties. For instance, a Carbonyl (CO) can be replaces with a (larger) Sulfonyl (SO_2) to increase the number of hydrogen bonds that can be formed. Bioisosteric exchange results in different molecules with the same biological properties. Another strategy is to make the molecule more rigid by introducing ring systems or to make it more flexible by adding aliphatic carbon chains. In the former case, the number of rotatable bonds is decreased, in the latter, it is increased. By increasing the lipophilicity, the absorption rate and affinity are usually increased as well.

In this phase, many computational tools are used since they enable researchers to conveniently create new molecules and quickly evaluate their properties. This includes calculation of physicochemical properties, analysis of conformational space, alignment of molecules, docking and scoring, or molecular dynamics simulation. Many of these require the 3D structure of the protein. If a structure is not available, statistical tools like quantitative structure activity relationship (QSAR) models can be applied. A QSAR model is trained with known experimentally determined or calculated properties and measured biological activities of a set of known molecules. It then estimate the activities of new molecules based on their properties. Statistical methods also play a role in the prediction of ADMET properties, i.e., how a compound is absorbed, distributed, metabolized and excreted concerning the organism and whether it is toxic. In addition, tools for the visualization of molecular interactions are used to inspect the outcome of the prediction or experiment. Finally, each interesting virtual molecule has to be synthesized and experimentally validated.

2.2.5 Pre-clinical and Clinical Testing

In this phase, the remaining compounds are tested *in vitro* (i.e. with cell cultures or isolated tissue cells) and *in vivo* (i.e. with mammals such as mice and dogs) for activity, bioavailability, and toxicity. Usually only a very small number of molecules or even just one passes these tests. Only then is the remaining molecule moved into the clinical phase, where it is tested on humans.

In phase I, a very small dose is administered to a small number of healthy volunteers to rule out toxicity and serious side effects. In phase II, the therapeutic effect and the proper dosage are determined. In phase III, the efficacy is demonstrated on a large scale to show that the new treatment is advantageous. Finally, the new drug is released. Only around 10% of the molecules that enter clinical trials become purchasable drugs [43].

2.3 Fragment-based Lead Discovery

In the last two decades, an alternative method to the traditional lead discovery process utilizing high throughput screening has evolved [44–49] and was established in pharmaceutical companies [23, 49]. *Fragment-based lead discovery* (FBLD) is based on very small molecules (fragments) rather than drug-sized molecules. The understanding of what qualifies as small, or *fragment-like*, differs slightly. For Carr et al. fragments are molecules with a molecular weight in the range of 120–250 Da [46], while Congreve et al. define a "Rule of three", which sets only an upper bound for the molecular weight to 300 Da [50].

The overall strategy of *fragment screening* is essentially the same as for high throughput screening with few differences in certain details. Fragment screening is usually done with a smaller library of molecules. Since the number of possible molecules rises exponentially with the number of atoms (molecular weight), there are fewer small molecules than larger ones. Therefore, a library of fragments covers a higher proportion of the available chemical space than drug-sized molecules [46]. Due to the smaller size of the molecules, the binding affinity that can be measured is much smaller. This is also an advantage since the binding affinity per atom is usually higher than for a non-fragment molecule hit [46, 48]. There are a number of alternative experimental techniques well suited for fragment-based lead discovery that differ in their throughput and resource requirements [46].

After one or several hits have been identified, these fragments can be combined to a drug-sized molecule. This gives medicinal chemists the opportunity to build a molecule with desirable drug properties, rather than try to modify an already large molecule [49]. A number of computational techniques were developed to support FBLD by finding appropriate linkers for these fragments [7, 22, 25]. Fragment-based computational methods will be discussed in more detail in chapter 4. First, the concept of chemical space as a means to describe interesting molecules is introduced in chapter 3. Within this context, it is discussed how fragments are utilized to model such space.

3 Chemical Space

During the drug development process, large amounts of molecules are encountered and reduced to the most promising entities. All methods employed in this field, whether experimental or computational, are indented to systematically search for molecules in this *chemical space*. The term is used ambiguously in the literature. In the most general sense, it refers to all possible molecules that could exist. However, this is better described by the term *chemical universe*, stressing the comprehensive and possibly infinite nature of this space. In the context of drug development, chemical space usually refers to "all the small carbon-based molecules that could in principle be created" [51]. A more precise description is provided by the same authors and was used to develop the following definition:

Definition 3.1. A **chemical space** is a multi-dimensional descriptor space, i.e., it is spanned by "a particular choice of [*molecular*] descriptors and the limits placed on them" [51].

These descriptors can be very concrete, as in "molecular mass" or "number of atoms"; or less concrete as in "all molecules that bind to a particular protein". Chemical spaces can be interpreted as subsets of the chemical universe and treated as mathematical sets; they can overlap, contain each other, or be disjoint. In drug development, one of the first steps is to get a feeling for how the chemical space of interest must be constituted, in other words, what properties molecules must exhibit to qualify as a ligand to a protein of interest. In the following, estimations for the size of interesting chemical spaces are discussed. Then, interesting chemical spaces are described based on their descriptors and respective thresholds. Finally, *combinatorial chemical space* as a basis for *de novo* design and *fragment space* as a model of combinatorial chemical space are introduced.

3.1 Size of Chemical Space

Molecules are of combinatorial nature. They are made of building blocks (atoms) that can be combined in various ways. Organic chemistry is mostly limited to six elements (hydrogen, carbon, nitrogen, oxygen, phosphorous, and sulfur), with carbon being the most common. All atoms have a limited set of valence states, i.e., electron configurations in which they can stably bind. Although the number of building blocks and their individual states are quite clear, the possibilities for combining these atoms are virtually infinite. Therefore, "finding" the one molecule with just the right properties is very difficult.

Several estimations for the size of the interesting chemical space exist. The most often referenced number is 10^{60} molecules, which is the result of a thought experiment constructing molecules with 30 atoms by Bohacek et al. [9]. Assuming an average of six atom types, 6^{30} ($\approx 2 * 10^{23}$) linear molecules exist. When taking ring closure and branching into account, there are $30 * \frac{28}{2}$ and 30^2 possibilities per molecule, respectively. This results in 10^{40} non-linear structures for each linear molecule and in a total of 10^{63} molecules. The authors conclude: ”Although this is a rough estimate, it seems likely that when all the different possible combinations of ring closure and branching are taken into account, the true number will be well in excess of 10^{60} and will rise steeply with increasing molecular weight” [9].

Walters et al. assume that ”there are perhaps millions of chemical libraries that a trained chemist could reasonably hope to synthesize” and that ”each library can, in principle, contain a huge number of compounds – easily billions” [10]. Based on this, the authors conclude that a ’virtual chemical space’ may contain 10^{100} molecules [10].

An estimation based on the analysis of an exhaustive set of known molecules was done by Drew et al. [11]. Therefore, the size of the chemical space is considered a function of the number of carbon atoms and was fitted to a power-function used for predictions. The authors estimate that the chemical space containing organic compounds with less than 100 carbon atoms contains $3.4 * 10^9$ molecules, a much lower number than the previous estimations. Furthermore, the *known drug space* and the drug-like chemical space are predicted to contain $2 * 10^6$ and $1.1 * 10^6$ compounds, respectively.

These examples show that the estimated numbers vary greatly depending on the parameters and models used. The most extensive database is the CAS registry of the Chemical Abstract Service (CAS), which contains ”more than 109 million

Table 3.1: Different estimations for the size of chemical space.

drug-like [11]	KDS [11]	Drew et al. [11]	Bohacek et al. [9]	Walters et al. [10]
$1.1 * 10^6$	$2 * 10^6$	$3.4 * 10^9$	10^{63}	10^{100}

unique organic and inorganic chemical substances" as of February 2016 [52]. Regardless of the estimation considered, it is obvious that only a tiny fraction of the chemical universe is known to date. Taking the most frequently cited number by Bohacek et al. as a basis, one can easily determine that only every 10^{52}th possible molecule is present in this database. Furthermore it becomes obvious that systematically generating all possible molecules is not feasible: Imagine a very fast computational method for generating molecules with a rate of 10^6 molecules per second ($\frac{mols}{s}$), that is run in parallel on a cluster with 10^6 nodes, thus working with an effective rate of 10^{12} $\frac{mols}{s}$. In order to enumerate all molecules estimated by Bohacek et al. a runtime of

$$\frac{10^{60} mols}{10^{12} \frac{mols}{s}} = 10^{48} s = 3 * 10^{40} years$$

must be expected. Suffice it to say that such an efficient method does not exist. Since it is impossible to systematically create all imaginable molecules, one must limit the search space to interesting parts of the chemical universe. In the next section, these chemical spaces are introduced.

3.2 Chemical Property Spaces

There are several chemical spaces that are interesting in drug development. They have been defined based on statistical analyses of known molecules regarding their properties. One of the most important qualities is bioavailability, i.e., whether a molecule can be absorbed by the human body and transported to its target location, often referred to as *drug-like*. Although there are several studies describing properties for drug-like molecules [13, 16–19], the most prominent and frequently referred one is the *Rule of five* (Ro5) by Lipinski et al., which is also referred to as "Lipinski's rule of 5" [15]. The name originates from the fact that the values are all integer multiples of five. It is based on simple and easy to compute physicochemical descriptors. The Ro5 states that molecules that exceed certain properties are less likely to be orally bioavailable, more precisely, that "poor absorption or permeation are more likely when" a molecule exhibits two or more of these properties [15]:

- molecular weight (MW) is over 500

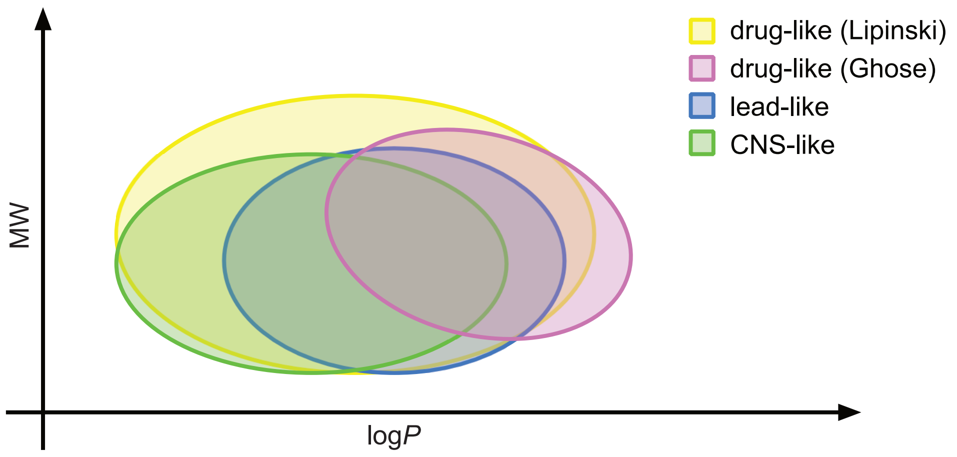


Figure 3.1: Artistic depiction of different chemical property spaces and their relation to each other with respect to molecular weight (MW) and solubility (log*P*).

- log*P* [2] is over 5
- more than 5 hydrogen-bond donors
- more than 10 hydrogen-bond acceptors

By inverting these values, one can define the drug-like space as a subset of the chemical universe. Hence, a molecule is considered drug-like if its properties do not exceed the given upper bounds. More stringent criteria are used to describe molecules that are more likely to cross the blood brain barrier and enter the central nervous system (*CNS-drug-like*) [17]. An extension to the *Rule of five* defines "qualifying ranges" of properties rather than single number thresholds [13]. A simpler description that uses only two properties is provided by Veber et al. [16]. Please refer to Table 3.2 for an overview. Ursu et al. ”summarize 18 papers focused on drug-likeness, and provide a comprehensive overview of progress in the field” [18].

As established in section 2.2, the first step in developing a drug is to identify lead structures. These molecules usually exhibit less molecular complexity than drugs to leave room for their optimization through bioisosteric exchange. According to Tudor Oprea and his coworkers, *leads* are not adequately described by the *Rule of five* [14]. The researchers, define a *lead-like* space that is described in several publications [12, 14]. The latter publication suggests that in the design of combinatorial libraries for lead discovery "care should be exercised not to exceed the following property values" [14]:

- 450 Dalton in MW

[2]log*P* is the octanol-water partition coefficient, a measure for the lipophylicity of a molecule.

- -3.5 <clog$P$ <+4.5
- 4 rings
- 10 nonterminal single bonds
- 5 hydrogen-bond donors
- 8 hydrogen-bond acceptors

Alternatively, Congreve et al. describe the *Rule of three* (Ro3) wich is used to describe lead-like or fragment-like properties [50] (see Table 3.2).

Physicochemical properties are often used to describe chemical space since they are part of the characterization of a molecule, easy to compute, and usually stored in a database for convenient access. However, more complex measures can be used as well. For instance, a Bayesian model trained on natural products to describe *natural product-likeness* [53]. In this way, many more chemical spaces can be defined, while the particular choice of descriptors and their range of values is usually determined by the specific use case. Although descriptors and their values may be well-defined, it is difficult to reliably predict how large these chemical spaces truly are. Therefore, all available boundary conditions are usually employed in order to find new molecules.

Table 3.2: List of sets of molecular properties that describe different bioavailable chemical spaces. (*) This number refers to the sum of H-bond donors and acceptors. Brackets denote alternatives.

	Drug-like Lipinski [15]	Drug-like Ghose [13]	Drug-like Veber [16]	CNS-like [17]	Lead-like [14]	Fragment-like [50]
MW	$\leq$ 500	160–480		$\leq$ 450	$\leq$ 450	$\leq$ 300
log$P$	$\leq$ 5	-0.4–5.6		$\leq$ 3	-3.5–4.5	$\leq$ 3
H-bond Donors	$\leq$ 5		[$\leq$ 12*]	$\leq$ 4	$\leq$ 5	$\leq$ 3
H-bond Acceptors	$\leq$ 10			$\leq$ 8	$\leq$ 8	$\leq$ 3
Rings					$\leq$ 4	
Atoms		20–70				
Non-terminal single bonds					$\leq$ 10	
Molar refractivity		40–130				
Rotatable bonds			$\leq$ 10			$\leq$ 3
TPSA			[$\leq$ 140]			

3.3 Combinatorial Spaces

A chemical space represents a set of molecules. As discussed before, it is not possible to create all molecules in order to subsequently search the interesting ones. Therefore, another way of modeling chemical space is required, which can be accomplished with a combinatorial approach. This concept is called combinatorial chemical space (short: *combinatorial space*) and is defined as follows [2]:

Definition 3.2. A **combinatorial chemical space** is a tuple of unique atomic or molecular *building blocks* and *connection rules.* Connection rules determine how building blocks relate and in which case a connection can be introduced.

Instead of storing millions of molecules in a database, only the building blocks and connection rules need to be stored. In this manner, large chemical libraries can be described efficiently [54, 55]. However, this poses a certain challenge when retrieving molecules and requires efficient algorithms for evaluating combinatorial spaces. To retrieve molecules, a graph representation of the space has to be created, in which the instance of a building block represents a node and edges are introduced based on connection rules. The manner in which connection rules are modeled and facilitated depends on the nature of the building blocks.

Several examples for combinatorial chemical spaces exist in nature, such as DNA and proteins. The "DNA space" consists of the four nucleotides as building blocks and the creation of phosphoester bonds as connection rules. For proteins, the building blocks are the 20 amino acids and amide bond formation represents the connection rule. Both spaces contain only a very small number of building blocks and a single connection rule based on a chemical reaction. Nevertheless, both spaces are huge because building blocks can in principle be combined in arbitrary order.

The first chemically relevant space was already mentioned at the beginning of section 3.1; it consists of atoms and their respective valence rules. However, since most atoms do not occur as single entities in nature, using this model may not result in synthesizable molecules. The largest class of computationally modeled combinatorial space is based on molecular fragments and connection rules that are inspired by chemical reactions. Peng reviewed several published combinatorial spaces of this kind "with sizes ranging from 10^{11} to 10^{20} compounds" [55]. Chapter 4 will discuss a variety of methods for modeling chemical space and for obtaining molecules from chemical space. Next, the fragment space model, which forms the basis of this thesis, is introduced.

3.4 Fragment Space

Fragment space is an elegant way to describe combinatorial chemical space and was developed by Matthias Rarey and Martin Stahl [24]. It is based on the preliminary work of Schneider et al. [56]. The fragment space data structure is part of the NAOMI library as well as its predecessor Flex*. It forms the basis for this project and numerous other methods [24, 25, 27, 29, 31, 32], which are described in more detail in section 4.3.4. Here, we will discuss the fragment space model.

The term *fragment* describes different entities depending on the context. It can mean an actual real life molecule, which is discussed in section 2.3, a model of such a molecule, and the data structure implementing the model. In the context of this thesis and the fragment space context in general, the following definition is used.

Definition 3.3. A **fragment** is a virtual molecule with at least one reactive site modeled as artificial atom, so called *link atom* or *linker*. A linker corresponds to an open valence, i.e., a position where a molecule can react with another molecule. A linker has a specific *link type*.

In terms of combinatorial space, fragments represent the building blocks of a fragment space (see Definition 3.2). The second component of a fragment space is the set of connection rules that describe the compatibility of link types. In other words, the connection rules define which link types are allowed to react with each other. A single connection rule consists of a pair of link types and a bond type. This information is used by an algorithm to determine which fragments can be connected. In principle, any string can be used to name a link type. In the examples presented in this thesis, per convention, link types are denoted by either letter L or R[3] followed by an integer, e.g., L1, R34, etc.

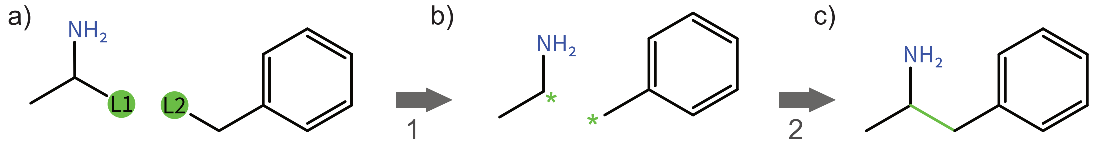
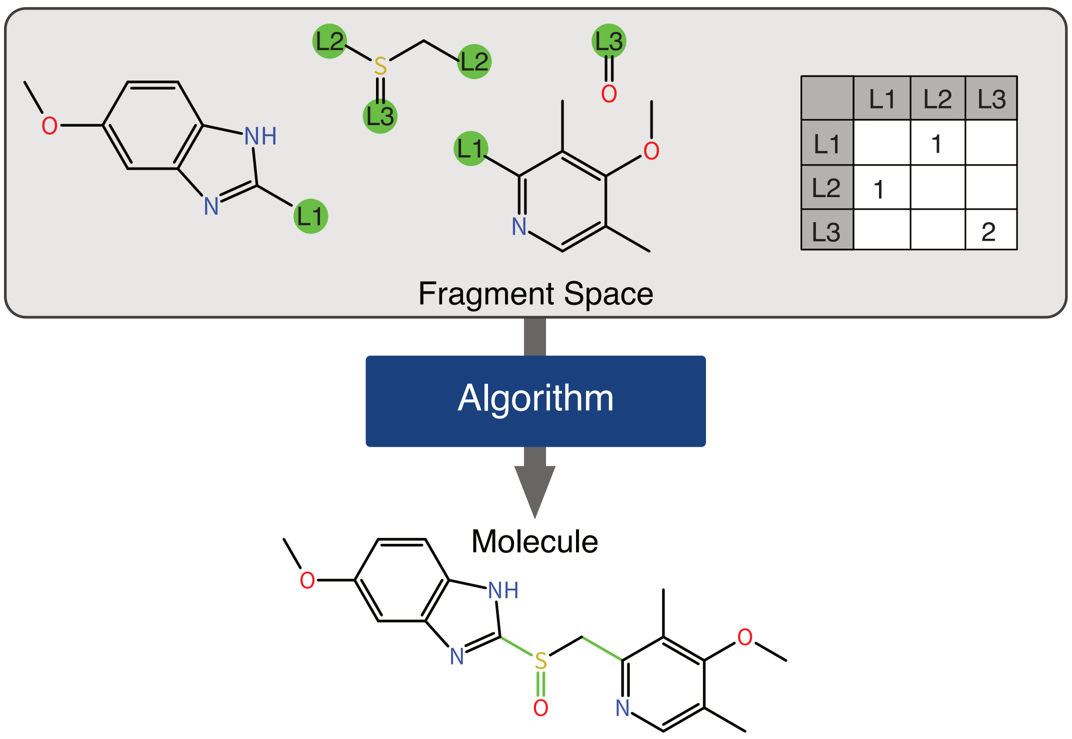


Figure 3.2: Depiction of the processing steps for connecting two fragments. a) Two fragments with one linker each are shown. Linkers are marked with green circles and decorated with link type L1 and L2, respectively. b) Intermediate state after processing step 1: Linkers and adjacent bonds are removed. The open valence at the reactive atom is highlighted with an asterisk. c) The resulting molecule after processing step 2: A new bond was added between the reactive atoms. The molecule is amphetamine (Drugbank DB00182).

The general procedure of creating molecules and the components of a fragment space are depicted in Figure 3.3. When two fragments are connected, the two compatible linkers that are involved in the connection are removed and a new

[3]In chemistry, R is used in structural formulae to denote a group that is attached to the *rest* of the molecule.

bond is added, thus connecting the two fragments in accordance with a connection rule (see Figure 3.2). This process essentially models a reaction that happens between two small organic molecules. Connection rules are derived from chemical reactions in order to facilitate synthesizable molecules. There are two ways this information is generated and used to construct a fragment space: Either via retrosynthetic fragmentation or forward-synthetic annotation. These processes are a principal subject matter of this thesis and are discussed in more detail in section 4.3 as well as in chapter 6.



	L1	L2	L3
L1		1	
L2	1		
L3			2

Figure 3.3: Depiction of the components of a fragment space and how a molecule is assembled from them. On the top, a fragment space with four fragments and the corresponding connection rule matrix are shown. Linkers are marked with green circles and their link type: L1, L2, or L3. The compatibility matrix defines which types of linkers can be connected and the bond type of the connection. On the bottom, a molecule that was assembled from the fragments above is shown. Newly formed bonds are depicted in green.

After several fragments have been connected, one or more open linkers may still remain. Since a valid molecule must not contain unsaturated linkers, i.e., open valences, a terminal fragment must be defined for each link type. These are usually very small fragments consisting of a single atom (e.g. hydrogen) or a very small functional group (e.g. methyl). Figure 3.4 shows two examples for both "unterminated" and "terminated" fragments.

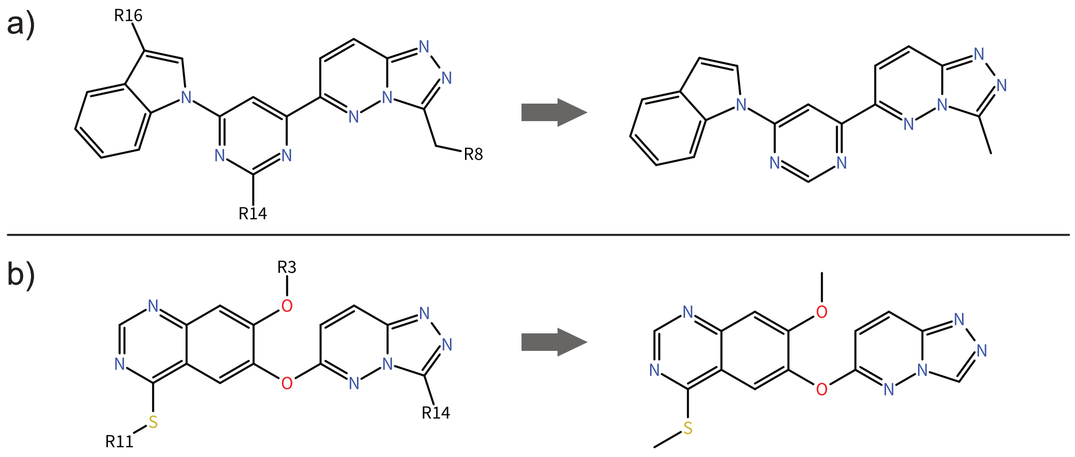


Figure 3.4: Two molecules constructed from fragment space fragments. a) and b) each depict the unterminated or chemically unsaturated molecule, on the left, and its terminated or saturated form, on the right. Termination was done based on BRICS rules [34]: R8, R14, and R16 are terminated with a hydrogen, R3 and R11 with a methyl group (see section 7.2.1)

Fragments are always connected via one bond only, thus forming a new single, double, or triple bond. Due to the history of fragment spaces, the formation of rings by creating ring bonds is not possible. The initial publication [24] used fragment spaces constructed with retrosynthetic rules of the Retrosynthetic Analysis Procedure (RECAP), which "cleave only acyclic bonds so that ring motifs are left intact" [57]. In addition, the feature tree descriptor [58], which is used for searching fragment spaces [24], uses rings as atomic functional units, i.e., as nodes.

3.4.1 Implementation

The fragment space data structure was first introduced in Flex* [24], the predecessor of the NAOMI cheminformatics library. NAOMI is developed in the research group of Matthias Rarey [59] in collaboration with BioSolveIT[4]. It implements a unique chemical model for representing molecules. A molecule is a graph with atoms as nodes and bonds as edges. In addition, several levels of chemical information are annotated to a single atom, i.e., element, valence state, and atom

[4]www.biosolveit.com

type [60, 61]. Bond types are annotated in accordance with the molecular environment, e.g., valence states. Most importantly, NAOMI includes functionality to initialize molecules from files in a reliable and deterministic way. The NAOMI library provides an application programming interface (API) to query this data structure. This forms the basis for a range of algorithms for the calculation of molecular properties and molecular descriptors (see chapter 5) as well as a variety of bio- and cheminformatics tools and methods [31, 32, 60–69].

The fragment data structure is essentially a wrapper for the NAOMI molecule extending it with functionality to annotate linkers as described in Definition 3.3. The molecule data structure natively supports linkers by providing custom element and atom type. Therefore, linkers can be read from and written to the supported file formats (i.e. SMILES, Mol2, SDF). The fragment data structure additionally stores a fragment ID to identify a fragment within the context of a fragment space.

The fragment space data structure of NAOMI holds a list of fragment instances and a compatibility matrix for link types. It provides basic functionality to add, retrieve, and delete individual fragments as well as connection rules. Besides the core data structure, the NAOMI library provides additional classes and modules to conveniently access and work with fragment spaces. This includes an index data structure that allows to efficiently query for lists of compatible fragments or fragments with a certain link type, functionality to read fragment spaces, and convenience functions to work with fragment space instances. The algorithms developed in this project became part of the fragment space library and the NAOMI code repository.

4 Strategies for Exploring Chemical Space

This chapter will give an overview of the different strategies used for the exploration of chemical space, i.e., the generation of (virtual) molecules. This includes a characterization of the different methods and a discussion of their advantages and disadvantages. There are numerous *de novo* design methods [7, 8, 54] that face the problem of finding the *right* molecule(s) in the vast chemical universe, i.e., molecules with ideal properties concerning bioavailability, specificity, and toxicity. The common strategy of all methods is to limit search space by applying constraints including physicochemical properties, molecular topology, three-dimensional shape, protein-structure, and chemical function. A given method may use any combination of constraints. In addition, the methods differ in various other aspects of their workings. They can be categorized according to their building blocks, primary target constraints, search strategy, structure sampling or scoring function as done by Schneider et al. [7]. Other categories are the type of algorithm (whether a stochastic or a deterministic strategy is used) and synthetic feasibility (whether the resulting molecule has a chance of being synthesized), with the former determining whether a chemical space is searched exhaustively or incompletely. The latter is the most crucial aspect and the biggest challenge for computational methods. Arbitrarily assembled atoms do not necessarily constitute valid molecules. *Valid* in this case means whether it is possible to create this molecule based on known organic chemistry. Therefore, synthetic feasibility should always be considered when generating molecules by computation.

In the following, methods are categorized according to their most significant feature with respect to the algorithms developed and implemented for this work.

This takes into consideration whether they are stochastic or deterministic, how they "navigate" chemical space, how they utilize combinatorial chemical space, and how they utilize chemical reactions. Furthermore, the underlying building blocks, i.e., the level at which modifications are performed, are incorporated as well. These are atom-, fragment-, or (molecular) graph-based. However, this is not a disjoint classification since methods may combine several strategies.

4.1 Stochastic Exploration

There are methods that are based on stochastic algorithms, such as evolutionary [53, 56, 70–75] or Monte Carlo [76] algorithms. The former is a class of optimization algorithms, which is inspired by biological evolution. It uses a fitness function, a set of modifications, and requires a starting point, usually a molecule with desired properties. The general strategy is as follows: Modifications are applied randomly to the input molecule, thus creating a set of new molecules, the first generation. Every molecule in this generation is evaluated by the fitness function. For each generation, the following steps are repeated until a termination criterion is reached. (1) The molecules with the highest fitness are selected and (2) randomly modified. (3) The fitness is calculated for all new molecules and (4) the least-fit molecules from the previous generation are replaced. Approaches differ mainly in the type of fitness function, the selection criterion, and the level at which modifications are applied.

Most methods are fragment-based [56, 70, 71, 74], while only a few are atom-based [53, 72]. Although atom-based modification may yield very promising theoretical molecules, they are not very realistic. Chemical reactions that exchange or add just a single atom are not common. Fragment-based modification resembles actual chemical reactions much more closely if chemical reaction information is modeled into the method. This will be discussed in more detail in section 4.3.

The types of fitness functions utilized range from physicochemical properties [53, 72] over molecular descriptor similarity [56, 74] to docking scores [71]. Some methods use more than one approach and/or combine several. An interactive method uses the knowledge of the user as fitness function [72]. All previously mentioned methods use a fitness function to calculate the similarity to a query. Yet, there is one method that uses a diversity function to produce a "representative universal library" rather than a set of similar molecules [73, 75].

Monte Carlo algorithms are randomized algorithms that accept wrong solutions with a small probability. Although the result may not be correct, Monte Carlo algorithms are usually much faster than deterministic methods. The algorithm can be run several times with independent random numbers in order to reduce the error probability. Hu et al. use a Monte Carlo approach based on quantum mechanical considerations rather than a combinatorial chemistry approach [76].

Stochastic algorithms limit the search space in addition to initial constraints by applying a certain amount of randomness. As a result, they do not consider all possible solutions and cannot exhaustively search chemical space.

4.2 Exhaustive Exploration

Structure elucidation algorithms represent a large class of deterministic methods for enumerating chemical space [77–83]. They were developed in the context of chemistry from the need to derive molecular structure from experimental data such as mass spectroscopy or similar techniques. The outcome of these experiments is essentially just the mass from which a chemical formula can be derived. Structure elucidation in mathematical terms is graph enumeration with constraints. For each chemical formula, one builds a (molecular) graph with nodes representing atoms. The valence of an atom defines the degree of the corresponding node and thus serves as a constraint. By systematically trying all possible bond configurations, all valid molecular isomers are enumerated. All methods allow to define additional constraints such as allowed or disallowed substructures. Substructures are parts of the molecular graph for which the bond configuration is known, e.g., a ring system or a functional group. This information can only be derived from knowledge about the experiment by an experienced scientist.

The problem of structure elucidation is considered a "logically straightforward task and, assuming an error-free coding, is 100% reliable" [80]. These methods work well for the purpose for which they were intended, i.e., for enumerating *small* molecules namely those encountered in mass spectroscopy. However, generating large molecular libraries with this approach is not feasible due to combinatorial explosion. Peironcely et al. show several examples of molecules that "were not generated due to excessive computational time needed to generate all the candidate structures" by the open molecule generator (OMG) [83]. Two of the chemical formulae that failed were $C_5H_4N_4O_3$ and $C_9H_{11}NO_2$, with 12 heavy atoms each. These molecules are not very big in comparison to actual drug molecules. Of the 1528 unique molecules in the *approved* set of DrugBank (Version 4.3) [84], only 10% (154) of the molecules have 12 or less heavy atoms.

4.2.1 Chemical Universe Database

A very ambitious project related to structure elucidation algorithms is the chemical universe database GDB. This project attempts to systematically create all organic molecules starting with the enumeration of graphs. The only constraints are an upper bound for the number of heavy-atoms and the fact that the set of elements used is limited. The dataset has been published for molecules with 11 (GDB-11), 13 (GDB-13), and 17 (GDB-17) heavy-atoms, yielding $1.3 * 10^7$, $9.7 * 10^8$, and $1.6 * 10^{11}$ molecules, respectively [85–87]. Molecules from the different iterations have been used in docking and virtual screening studies [88–94]. The following description briefly summarizes the most recent approach, GDB-17.

As elements carbon, nitrogen, oxygen, sulfur, and halogens are considered. The process consists of four steps. First, a graph enumeration algorithm (GENG, [95]) is used to enumerate all graphs with up to 17 nodes. The degree of a node is limited to four, i.e., the maximal valence of a carbon atom. Second, graphs are translated into hydrocarbon molecules by replacing nodes with carbon atoms. Third, the single bonds in the hydrocarbons are systematically substituted with double, triple, and aromatic bonds thus creating so called *skeletons*. Finally, skeletons are subject to a diversification step that replaces suitable carbon atoms with oxygen and nitrogen atoms. Furthermore, halogens are added at distinct positions and functional groups are annotated. In each step, filter rules based on organic chemistry are applied to remove unsuitable structures. This process yields $1.6 * 10^{11}$ molecules, which were then compared to molecules with up to 17 atoms from other databases. The authors found that "57% of PubChem-17, 60% of ChEMBL-17, and 68% of DrugBank-17 are compatible with the GDB-17 enumeration rules" [87]. GDB-17 represents a specific area of the chemical space to a large degree and probably contains a lot of promising lead structures. However, many of these molecules may not be synthesizable. Furthermore, the majority of the molecules of interest contain more than 17 atoms. The "Approved Drugs" set of DrugBank [84] contains 1713 unique molecules of which 1286 (75%) have more than 17 heavy atoms[5].

4.3 Reaction-based Exploration

The biggest reservation of medicinal chemists towards computational methods is the – for the most part – theoretical nature of molecules. Therefore, the ultimate goal is to design molecules that can be synthesized in the laboratory immediately. Incorporating information about chemical reactions to suggest synthetic routes will hopefully improve the acceptance of computational tools. In order to improve the quality of the outcome, synthetic reactions selected for application in computational methods should be widely applicability and have high yields [96].

Although there are fragment-based methods that do not consider reaction information [70, 71, 74], all reaction-based methods use fragments. Reaction-

[5]Analysis based on DrugBank 4.3, filtered with MONA [63, 66]

and fragment-based methods are naturally intertwined because organic reactions require (small) molecules as reactants.

4.3.1 Combinatorial Libraries

A simple and widely used method to describe combinatorial space utilizing reactions is a combinatorial library. It is the virtual equivalent of a combinatorial chemistry experiment, where two types of reactants are mixed. A library is limited to a single reaction (connection rule) and contains two sets of preselected, compatible molecules (building blocks). For example, for two sets A and B, each element A_n can, in principle, react with each element B_m: $\mathrm{A_n + B_m \longrightarrow C_{nm}}$. Thus, the number of molecules represented by the library is $|A| * |B|$.

There are a number of different tools utilizing combinatorial libraries, that each bear their own advantages and limitations. The first examples are MoSELECT [97] and AutoClickChem [98], which both apply the reaction to the reactants of a combinatorial library. MoSELECT uses a genetic algorithm that explores multiple objectives (e.g. several physicochemical properties) simultaneously to find product molecules with desired properties. AutoClickChem [98] uses only reactions from click chemistry, i.e., reactions that are fast, cheap, and comparatively easy to carry out in the laboratory. The advantage of this method is that it explicitly takes the 3D configuration of the reactants into account and uses it when a new molecule is constructed. Yet, the output molecules are not optimized towards their properties. Since the input for both methods is already tailored to the reaction, the synthetic protocol for all resulting molecules is the same. Both methods cover a relatively small chemical space because they are limited to one reaction at a time. However, through repeated application of different reactions to intermediate products, they could in principle be used to generate a larger number of molecules. Methods that natively apply multiple reactions will be discussed in sections 4.3.2 and 4.3.3.

4.3.2 Explicitly Modeled Reactions

This type of algorithm implements the repeated application of reactions inherently. Information about specific reactions must be provided explicitly as input. In each iteration, the reactive atoms are identified in each molecule. After a selection process, the reaction is applied to compatible reactants. The result is a sequence of synthesis steps that lead to the construction of a molecule with desired properties.

The method by Schürer et al. [99] as well as the tools SYNOPSIS [20] and DOGS [21] use explicitly modeled reactions. Schürer et al. use a genetic algorithm with a user defined fitness function. Reactions are provided in the SMIRKS reaction transform language [100]. The algorithm uses an iterative scheme that translates input molecules and intermediate products into combinatorial libraries, which is evaluated by the genetic algorithm to generate products in compliance with desired properties. SYNOPSIS (synthesize and optimize system *in silico*) is a stochastic approach which utilizes a user-defined fitness function similar to a genetic algorithm [20]. Molecules are chosen for extension according to the fitness function, while reactions and reaction partners are chosen randomly. DOGS (design of genuine structures) is closely related to SYNOPSIS and is a deterministic method that iteratively assembles fragments into new molecules by completely enumerating all possible solutions in each step [21]. For each initial molecule and intermediate product, all compatible reactions and all compatible reactants are used. A greedy strategy is applied that allows only for the best scoring products to advance. This behavior renders the method deterministic but it not exhaustive. The output of both methods is not only molecules, but also a list of reactants and the reactions used to join them. In the accompanying study to SYNOPSIS, "18 of the 28 designed molecules could readily be synthesized, and 10 of the synthesized molecules showed [...] inhibitory activity *in vitro*" [20]. DOGS has been applied in a prospective study and "following the proposed synthesis route, the [selected] compound was accessible and found to have the desired biological effect and selectivity profile *in vitro*" [21].

A recent method utilized reaction information to build a database of product molecules named SCUBIDOO (Screenable Chemical Universe Based on Intuitive Data OrganizatiOn) [101]. In order to construct this database, 58 commonly used, robust reactions [96] were applied to 18,561 common molecular building blocks [102]. The building blocks were first filtered down to 7805 molecules by imposing the following constraints:

- molecular weight ≤ 250
- rotatable bonds ≤ 2
- number of chiral centers ≤ 1

All pairs of molecules that are compatible regarding a certain reaction were connected based on the bond configuration defined by this reaction. This yields 2.110^7 new molecules after generation of stereoisomers. The database is freely accessible and provides "three representative samples of different sizes (S, M, and L)" [101].

4.3.3 Implicitly Modeled Reaction

In the case of implicitly used reactions, a preprocessing step to construct a combinatorial chemical space is required. Here, the knowledge about synthetic reactions is folded into this space and thus not explicitly considered when retrieving molecules. There are two options for the construction of such a combinatorial space, i.e., using reactions in a forward- or retro-synthetic way.

The latter uses known, larger molecules which are cleaved into fragments at specific bond types. These bonds were identified to originate from common chemical reactions by retrosynthetic analysis [34, 57]. During fragmentation, the bond type information is attached to the cleaved site via a *marker* or *linker*. This marker denotes reactive compatibility. The two prominent methods are RECAP (retrosynthetic combinatorial analysis procedure) [57] and BRICS (breaking of retrosynthetically interesting chemical substructures) [34] defining 11 and 16 bond types, respectively. Since BRICS plays an important role for this work, its strategy is discussed in chapter 6. A comparison of both methods is given in section 7.2.

The RECAP approach is used by several tools that utilize combinatorial space to generate novel molecules. A number of tools using stochastic algorithms has been developed by Gisbert Schneider and his coworkers [56, 103–105] following a similar scheme. First, RECAP is applied to generate a library of fragments. Then, a stochastic algorithm is applied to generate new molecules. TOPAS (topology assigning system) uses a template structure as a starting point for its evolutionary procedure [56]. The input structure is cleaved based on the retrosynthetic rules defined by RECAP. Individual fragments are substituted by randomly selected, compatible ones from the library, yielding 100 newly generated structures. These molecules are selected for further modification based on one of two possible fitness functions, either 2D-structural similarity or topological pharmacophore distance. FLUX (fragment-based ligand builder reaxions) is an advancement of TOPAS employing a similar evolutionary scheme [103]. Other than in TOPAS, input and intermediate molecules are fragmented based on a randomly selected rule from RECAP and all possible molecules are generated by utilising the fragment library. To assess fitness, two descriptors can be used with either Tanimoto similarity or Euclidean Distance (see section 5.2.1). In a second publication, different mutation and crossover operators were evaluated [104]. COLIBREE (combinatorial library breeding) uses a different approach as the algorithm is also of stochastic nature, but uses particle swarm optimization for generating new molecular structures [105]. The initial scaffold with several linkers remains constant during optimization. At the linker positions, different building blocks from the combinatorial space are attached as side chains. The fitness of a product molecule is assessed based on pharmacophore similarity to reference ligands.

The second option for folding chemical knowledge into a combinatorial space is similar to the explicit case discussed earlier. It uses reaction information for forward-synthesis. Starting from small molecules, their reactive groups are replaced by markers specific to a synthetic reaction. The difference is that the information about reactions is only required during preprocessing. The type of algorithm that utilizes the resulting combinatorial space is irrelevant at this point. This approach is accessible via the commercial tool CoLibri (compound library toolkit) [106]. It was used to construct the freely available KnowledgeSpace[6] [35] and the in-house space BI-CLAIM (Boehringer Ingelheim Comprehensive Library of Accessible Innovative Molecules) [107]. The former was assembled from 82 reactions, contains 10,879 fragments, and may describe as many as $1.2 * 10^{10}$ molecules. Nikitin et al. describe the construction of a combinatorial space "from about 400 combinatorial libraries" [108]. The authors claim that this space contains "more than 10^{13} chemical compounds" [108]. Peng reviewed eight methods for constructing virtual compound spaces [55], several of which are combinatorial spaces based on reaction information.

The result of both retrosynthetic fragmentation and (forward) synthetic annotation is a set of fragments that can be combined in new ways based on the rules utilized in the build process. Section 4.3.4 gives an overview of existing methods utilizing a concrete implementation of such combinatorial spaces.

4.3.4 Fragment Space-based Methods

This section summarizes existing methods for construction of fragment spaces as well as using them to construct new molecules. Figure 1.1 gives an schematic overview of how these methods relate.

In the past, fragment spaces were used to describe both retrosynthetic [24, 34] and forward-synthetic [35, 106] combinatorial space. BRICS [34] was developed in the fragment space context as an advancement of RECAP [57]. Since the construction and search of fragment space are decoupled, a number of methods for retrieving molecules were developed. FTreesFS is an algorithm for similarity searching in fragment spaces [24]. Rather than searching this space in a stochastic fashion, the algorithm deterministically searches all molecules with a user-defined similarity regarding a query structure. It uses the *Feature Trees* descriptor [58], which is well suited for the fragment-based context.

FlexNOVO is a structure-based tool for ligand design [25]. It uses fragments from a fragment space to sequentially grow a molecule and the docking algorithm FlexX to place and score the fragments. Recore is a tool for scaffold replacement [27]. Based on a query ligand with 3D-structure, it replaces the core of this molecule with fragments from a fragment space. During the construction of the fragment space, 3D coordinates are conserved so that they can be used by Recore. SmartsFS implements a substructure search [31, 32] based on the SMARTS pattern language [100]. Other than search algorithms, a method for the construction

[6]http://www.biosolveit.de/FTrees-FS/

of combinatorial libraries based on fragment spaces, LoFT, exists [109]. Since the resulting combinatorial libraries can be output as fragment spaces, they can subsequently be subjected to available search algorithms or enumeration.

There also is a method for the systematic enumeration of fragment spaces by Pärn et al. based on the Flex* library [29]. The only constraints this method uses are physicochemical properties, thus potentially very large molecular libraries may be created. However, the algorithm relies on the main memory of a computer and is therefore limited in the number of molecules that can be generated. As a result, very stringent constraints were used for the enumeration and relatively small libraries were generated (21 – 779,213 molecules). This method is discussed in more detail in section 6.3.

4.4 Unsolved Challenges

The algorithms and tools for the exploration of chemical space all have advantages and disadvantages. The first group of methods cannot exhaustively enumerate a chemical (sub)space because they use stochastic algorithms [53, 56, 70–76]. They are intended to design molecules that are similar to a query structure. The chemical space is usually very localized and restricted by different molecular descriptors (see definition 5.1) and the choice of similarity measure (see definition 5.2). Due to the utilization of random numbers, these algorithms are not guaranteed to find the best solution, or all possible solutions.

The second group of algorithms is designed to exhaustively explore chemical space [77–83]. These methods are structure elucidation tools intended to generate all possible molecules that are described by a chemical formula. They were designed to aid the evaluation of experimental data describing small molecules. They work very well for the intended usage, but struggle with larger structures, i.e., molecules with 12 atoms and more. A special case in this group is the Chemical Universe Database [85–87], which contains a large number of molecules that only contain 17 or less atoms.

The last class of methods discussed incorporates reaction information and utilizes a range of algorithms [20, 21, 56, 97–99, 101, 103–106]. Some exhaustively enumerate combinatorial libraries, only covering a single reaction [97, 98] (section 4.3.1). Others apply chemical reactions directly to molecules (section 4.3.2), but use a stochastic algorithm [20, 99], a greedy strategy [21], or are limited to one reaction step [101]. The last group builds combinatorial spaces [25, 31, 32, 35, 55, 56, 103–109] that can subsequently be evaluated by different algorithms. This includes many algorithms using stringent constraints such as molecular similarity – while both stochastic [56, 103–105] and deterministic [24] methods exist – or 3D-structural constraints [25, 27]. By incorporating reaction information, the likelihood of being able to synthesize molecules is increased [20, 21]. In this context, the concept of fragment spaces was introduced [24], which is also applied in this project.

To sum up, most algorithms do not enumerate chemical space exhaustively due to their stochastic nature or application of very stringent constraints. Deterministic approaches, on the other hand, only enumerate small subspaces because they also apply stringent constraints. These methods are mostly used in later stages of the drug development pipeline, when certain properties about the target molecule and possible ligands, i.e., specific constraints, are already known. In earlier stages, these properties may not be known thus larger chemical space must be investigated. Methods for generating large diverse sets of molecules with less specific constraints are required. That is why in this work, the problem of exhaustive enumeration of large chemical space is addressed. Another important aspect that is often ignored is whether generated molecules are synthesizeable. A lot of methods apply unrealistic modifications to molecules neglecting information about chemical reactions. In order to increase the likelihood that generated molecules can be synthesized, in this work, reaction-based modifications are employed. In particular, a method for reaction-based construction of fragment spaces was developed. By utilizing fragment spaces, the construction and evaluation of combinatorial space is decoupled, thus allowing to address one problem at a time.

5 Modeling Chemical Information

Molecules are the central body of interest in cheminformatics and drug discovery. In order to work with them, they must be transformed into a numeric representation that allows for mathematical and algorithmic processing. This representation is a model of the actual molecule, i.e., an interpretation of the physical world. This chapter addresses the following questions. How can chemical information, especially molecules and chemical property space, be modeled? And, how can the quality of a generated molecular library be assessed based on these models?

With respect to the latter, the answer is not that easy. Ultimately, designed molecules should be synthesized and experimentally characterized. However, this is not feasible and not possible in many cases. The straightforward approach would be to subject them to the virtual drug-discovery pipeline (see section 2.2), i.e., virtual screening, consecutive filtering steps, QSAR analysis, and other methods. Yet, setting up individual screening campaigns and further analysis is not within the scope of this thesis. It is time consuming, requires setting up many additional experiments and, to do it right, requires expert knowledge. In order to evaluate the results generated, an easy to apply, fast to compute, objective, and reproducible approach based on molecular similarity was used (see section 8.2.3). Generated molecules are compared with existing, well characterized molecules and assessed based on their pairwise similarity.

In this chapter, the concept of molecular descriptors as a means to describe molecules and their properties is discussed (section 5.1). This includes the introduction of descriptors suited for the comparison of large chemical libraries and measures to determine the similarity based on these descriptors (section 5.2).

The methods presented here were implemented in the NAOMI library, but are discussed independent of an implementation. The implementation of the NAOMI molecule model and the tools developed in this project are discussed in chapter 9.

5.1 Molecular Descriptors

A great number of different *molecular descriptors* (short: *descriptors*) containing different types of (physico-)chemical information about a molecule exist [110]. They are used to condense complex information of a molecule into a compact format to make it accessible for efficient and simple comparison, e.g., to quickly check whether a molecule fulfills certain criteria regarding solubility, molecular weight, or other physicochemical properties. Others are used to describe a molecule in great detail to enable exchange of information, e.g., file formats for molecular information. Todeschini and Consonni define the term as follows:

Definition 5.1. "The molecular descriptor is the final result of a logic and mathematical procedure which transforms chemical information encoded within a symbolic representation of a molecule into a useful number or the result of some standardized experiment." [110]

This definition thus divides molecular descriptors into two categories – experimental and theoretical – thus differentiating between the way they were obtained. *Experimental measurements* are derived from experiments in standardized laboratory conditions. They are physicochemical properties, e.g., $\log P$, dipole moment, or polarizability, or even more complex information such as 3-dimensional structure. *Theoretical descriptors* are computed by an algorithm, based on a model of a molecule. Some of these computational methods predict the same physicochemical properties as measured in an experiment. The advantage is that they can be calculated much faster and be computed for theoretical molecules. In the latter case, they can predict properties before a molecule is synthesized.

The usefulness of theoretical descriptors arises from their flexibility. They allow for diverse information to be connected and combined to illuminate properties that cannot be determined via experiments, e.g., topological properties as defined by the extended connectivity fingerprint (ECFP) descriptor [111]. Furthermore, they can be used for comparison since they are calculated in a standardized way.

Molecular descriptors can be classified according to the dimensionality of the information they contain:

0D single value (e.g. number of hydrogen-bond donors, logP)

1D list of values, linear data (e.g. fingerprints, ECFP [111], Simplified Molecular Input Line Entry Specification (SMILES) [100])

2D matrix of values, tabular data (e.g. graph representation of a molecule)

3D 3-dimensional data (e.g. spatial representation of molecules, surface or volume descriptors)

4D 3D plus supplementing information as 4th dimension (e.g. conformational space)

In the enumeration algorithm that was developed in this project (see 6.4) 0D-descriptors are used to describe chemical space. They represent those physicochemical properties most often used to characterize molecules (see 3.2) and can be efficiently compared to thresholds. Many properties are trivially computed by counting, e.g., number of atoms, number of non-hydrogen atoms and bonds, or molecular mass. Some properties are derived from molecular topology, e.g., rotatable bonds, stereo centers, or hydrogen-bond donors and acceptors. There are also properties for which more sophisticated algorithms are required, e.g., the computation of solubility as $\log P$ [112]. All computations are based on the NAOMI molecule data structure and are readily accessible via NAOMI functions.

In addition, 1D-descriptors are utilized in two cases. The first is duplicate checking. During the enumeration algorithm introduced in chapter 6, molecules are compared in order to determine whether a newly generated molecule is identical to a previously encountered molecule. A simple binary decision has to be made, whether two molecules are identical or not. For this, the molecular line notation SMILES [113, 114] is used. The functionality to compute a SMILES descriptor for a molecule is provided by the NAOMI library. The second application is the comparison of large sets of molecules. This is required in chapter 8 in order to analyze the results. Since very large numbers of molecules are encountered, it is required to do this efficiently. In this project, topological fingerprint descriptors are used for this purpose, This type of descriptor is discussed next.

5.1.1 Fingerprint Descriptors

Molecules have to be available in a well-defined numeric form in order to be compared. The most precise comparison would be via molecular graphs. However, since the determination of graph isomorphism is in NP [115, 116], this is not a desirable approach. In addition, graph isomorphism is not necessarily suited to determine the level of similarity between molecules since it does not encompass chemical information. For this purpose, molecular descriptors are much more appropriate. How the level or amount of similarity is quantified will be discussed in section 5.2.1. First, a suitable descriptor will be discussed.

From the vast amount of available descriptors [110], only a few are used for similarity comparison [117, 118]. Fingerprint descriptors are well established for this purpose since they are easy to compute and efficient to store. A fingerprint or *feature vector* consists of a fixed-size bit-array in which every position denotes whether a certain feature, such as a functional group, is present (1) or not (0). Comparison of these fingerprints is very efficient since merely two equally sized bit arrays must be compared. Fingerprints are sometimes criticized because they do not account for how often a feature occurs in a molecule. This is taken into account by integer fingerprints. They are modeled just as binary fingerprints, but several bits are used to describe a feature instead of just one bit [118].

A further critique of fingerprints is that they are limited to a fixed number of features. A solution to this is to store the indices of the encountered features rather than using a fixed-size array. This approach is used by the Extended-connectivity fingerprint, the functional-class fingerprint (FCFP) [111], and the topological torsions descriptor [119]. These fingerprints do not use a fixed set of predefined features, but rather define a function to calculate a feature based on the properties of a molecule. This function combines the values of the individual properties and maps them onto a finite number range by means of a hash function, e.g., 32- or 64-bit integer. The resulting numbers are stored in a vector or set. In this way, only the indices of the features present are stored and used in the similarity calculation. On the one hand, the number of features is still limited to 32- or 64-bit integer, on the other hand this means that there are $2^{32} \approx 4.2 * 10^9$ and $2^{64} \approx 1.8 * 10^{19}$ possible features, respectively.

Extended-connectivity fingerprints have been developed around the year 2000 [111]. In 2010, Rogers et al. published the details of the algorithm, listing more than 80 publications in which ECFPs are used successfully [111]. FCFPs were introduced in the same publication and operate on the same principle. ECFPs take information about molecular topology into account by incorporating neighboring atoms. First, for each atom, the binary values of the following properties are combined into a single number: "the number of immediate neighbors who are 'heavy' (non-hydrogen) atoms; the valence minus the number of hydrogens; the atomic number; the atomic mass; the atomic charge; and the number of attached hydrogens (both implicit and explicit)" [111] and "whether the atom is contained in at least one ring" [111]. The resulting number is subjected to a 32-bit hash function, thus creating the *atom identifier*, which is subsequently added to the identifier set (see Figure 5.1 a). Then, the iterative update step is carried out a user-specified number of times. Each atom identifier is combined with the identifiers of the neighboring atoms from the previous iteration and then hashed (see Figure 5.1 b). The resulting atom identifier is added to the identifier set. This way, in each iteration a larger environment of the respective atom is folded into the current identifier. Thus, atoms having the same environment will have the same atom identifier. After the iteration phase has ended, the current identifier set represents the ECFP.

FCFPs are computed identically to ECFP except for the use of initial descriptors. They use "a more abstract, pharmacophoric set of initial atom identifiers [...]. Each atom is identified by a six-bit code, where a given bit is 'on' if the atom plays the associated role. The atom roles are: hydrogen-bond acceptor and donor; negatively and positively ionizable; aromatic; and halogen" [111].

The bond diameter d of both ECFP and FCFP is a function of the number of iterations i:

$$d(i) = i * 2$$

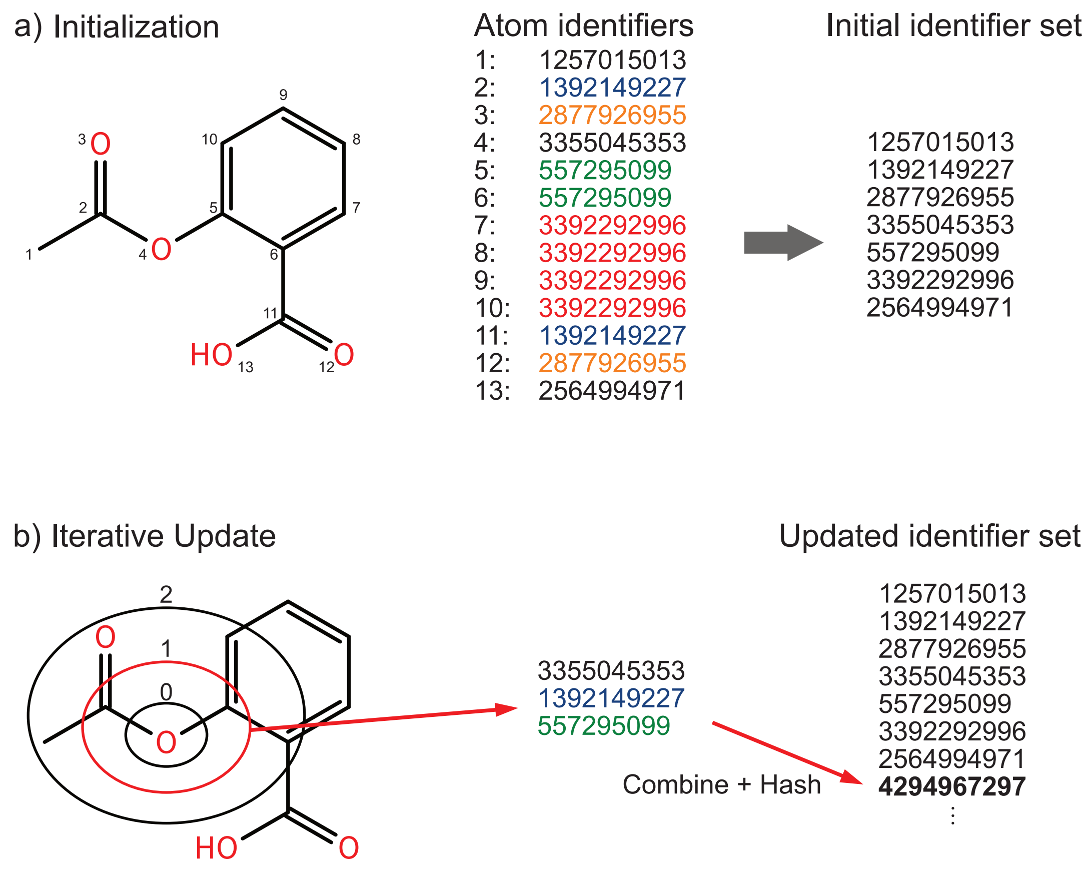
The number of iterations "depends on the desired use of the fingerprint" [111]. To specify the diameter, Rogers et al. suggest the following convention (see Figure 5.1 c): "[name of the] fingerprint using a four-character string (e.g., "ECFP"), followed by an underscore, followed by a number" describing the effective diameter [111], e.g., ECFP_4 for a diameter of four.

Both ECFP and FCFP are well suited for use with molecules constructed from fragments since the substructure that a fragment introduces to the molecule leads to the same fingerprint, regardless of orientation within the molecule. However, different identifiers will be assigned to the atoms at the borders of the fragment, i.e., at the connection points, naturally, accounting for the specifics of the new molecule.

Another related fingerprint descriptor uses *topological torsions* (TT) [119]. A "topological torsion [is] a linear sequence of four consecutively bonded non-hydrogen atoms, each described by its atomic type, the number of non-hydrogen branches attached to it, and its number of π electron pairs" [119]. Similar to the ECFP approach, the information about each atom is transformed into a 32-bit word, while "each atom-type field occupies 4 bits; the number of π electrons and the number of non-hydrogen-atom branchings each occupy 2 bits. To prevent the same TT type from being coded in two different ways, a canonical packing scheme is used." [119]. Analogous to the ECFP, each of the generated 32-bit integers are added to a set, which represents the TT fingerprint. The TT descriptor can be schematically described as follows:

```
(NPI-TYPE-NBR)-(NPI-TYPE-NBR)-(NPI-TYPE-NBR)-(NPI-TYPE-NBR)
```

In the end only ECFP was used for evaluating experimental results (see chapter 8). A brief validation study for the ECFP implementation is shown in Appendix C.



Atom identifiers

1:	1257015013
2:	1392149227
3:	2877926955
4:	3355045353
5:	557295099
6:	557295099
7:	3392292996
8:	3392292996
9:	3392292996
10:	3392292996
11:	1392149227
12:	2877926955
13:	2564994971

Initial identifier set

1257015013
1392149227
2877926955
3355045353
557295099
3392292996
2564994971

3355045353
1392149227
557295099

Updated identifier set

1257015013
1392149227
2877926955
3355045353
557295099
3392292996
2564994971
4294967297
⋮

c)

Iteration	Bond Diameter	Designation
0	0	ECFP_0
1	2	ECFP_2
2	4	ECFP_4
3	6	ECFP_6
⋮	⋮	⋮

Figure 5.1: Visual representation of the algorithm for Extended-connectivity fingerprints computation using the example of acetylsalicylic acid. a) Initial computation of identifiers: For each atom denoted in the molecule (left), an initial identifier is calculated (middle). Several atoms have the same environment and thus the same identifier as denoted by color. The unique set of initial identifiers represents the initial fingerprint, ECFP_0 (right). b) Depiction of the relationship between iteration and diameter. The numbers in the molecule denote the iteration.

5.2 Molecular Similarity

This is either a similarity or a distance coefficient [117]. While molecular descriptors encode the chemical information, these measures determine the level of similarity. In the following, the molecular descriptors used for similarity assessment and similarity measures are discussed. Although none of these descriptors were newly developed, they were implemented for use with the NAOMI library during this project.

5.2.1 Similarity Measures

The descriptors capture which chemical features are present in the molecule. In order to quantify the similarity of two molecules, a metric is required that converts this information into a single number. These measures are not limited to molecules, but can be used to determine (dis-)similarity of various objects, given a numeric representation. As input, these methods require two lists of numeric values, such as bit arrays or sets of integers. In practice, one differentiates two types of measures: similarity and distance coefficients.

Similarity coefficients measure similarity. These measures can be and mostly are normalized to the range of $[0.0, 1.0]$. The maximum value (1.0) denotes identity. A simple definition is given by Maggiora et al.:

Definition 5.2. "Similarity measures, also called similarity coefficients or indices, are functions that map pairs of compatible molecular representations that are of the same mathematical form into real numbers usually, but not always, lying on the unit interval." [118]

Distance coefficients measure dissimilarity and are often analogous to distance in geometrical space. The value for identity is 0 and the higher the value, the more dissimilar the molecules. For a distance coefficient D to be a *metric* in a mathematical sense, it must exhibit the following properties [117]. Values must be:

1. non-negative $D(x, y) \geq 0$
2. zero for identical objects $D(x, x) = D(y, y) = 0$
3. greater than zero for nonidentical objects $A \neq B \Leftrightarrow D(x, y) > 0$
4. symmetric $D(x, y) = D(y, x)$
5. and obey the triangular inequality $D(x, z) \leq D(x, y) + D(y, z)$.

Table 5.1 lists measures that are most frequently used in cheminformatics [117]. It contains the general formula and a set-theoretic definition for each measure. The former is intended to be used with bit array-like fingerprints, where values are either 0 or 1. In this case, vectors a and b are defined as follows: $a = (a_i)_{1\leq i\leq n}$ and $b = (b_i)_{1\leq i\leq n}$, respectively. However, in the case of ECFP and related fingerprints, the fingerprint vector contains integer values that represent the indices of the bits set to 1. In this case, the set-theoretic definition is applicable, where A and B represent sets of indices; $A = \{i : a_i = 1\}, B = \{i : b_i = 1\}$.

Table 5.1: List of frequently used similarity coefficients in cheminformatics. a and b denote vectors ($a = (a_i)_{1\leq i\leq n}, b = (b_i)_{1\leq i\leq n}$); A and B sets ($A = \{i : a_i = 1\}, B = \{i : b_i = 1\}$).

Name		
	general formula	set-theoretic definition
Tanimoto or Jaccard coefficient		
	$T_{a,b} = \frac{\sum_{n=1}^{n} a_i b_i}{\sum_{n=1}^{n} a_i^2 + \sum_{n=1}^{n} b_i^2 - \sum_{n=1}^{n} a_i b_i}$	$T_{A,B} = \frac{\lvert A \cap B\rvert}{\lvert A \cup B\rvert}$
Dice coefficient		
	$D_{a,b} = \frac{2\sum_{n=1}^{n} a_i b_i}{\sum_{n=1}^{n} a_i^2 + \sum_{n=1}^{n} b_i^2}$	$D_{A,B} = \frac{2\lvert A \cap B\rvert}{\lvert A\rvert + \lvert B\rvert}$
Cosine coefficient		
	$C_{a,b} = \frac{\sum_{n=1}^{n} a_i b_i}{\sqrt{\sum_{n=1}^{n} a_i^2 \sum_{n=1}^{n} b_i^2}}$	$C_{A,B} = \frac{\lvert A \cap B\rvert}{\sqrt{\lvert A\rvert\lvert B\rvert}}$
Hamming distance		
	$H_{a,b} = \sum_{i=1}^{n} \lvert a_i - b_i\rvert$	$H_{A,B} = \lvert A \cup B\rvert - \lvert A \cap B\rvert$
Euclidean distance		
	$E_{a,b} = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$	$E_{A,B} = \sqrt{\lvert A \cup B\rvert - \lvert A \cap B\rvert}$

6 Fragment Space Algorithms

In this chapter, three methods are discussed. They are organized in the order of how they are used in a fragment space workflow. First, two algorithms for the construction of fragment spaces are introduced. The first is based on retrosynthetic fragmentation and was developed by Degen et al. [34]. For this project, it was implemented based on the NAOMI cheminformatics library [59] and is used for data generation in numerous experiments. The second represents a newly developed, fully automated approach to construct fragment spaces based on synthetic reactions. Fragment spaces that were constructed in this manner are much more likely to produce synthesizable molecules. Second, an algorithm for the enumeration of molecules from fragment spaces is discussed. This newly developed method is based on the NAOMI library. It uses a deterministic approach to systematically generate all molecules with a user-defined physicochemical profile. In this manner, it can exhaustively construct all molecules in a certain chemical space.

6.1 Fragmentation

During fragmentation, a set of molecules is cut at specific bonds and thus converted into fragments (see Figure 6.1). This approach is analogous to retrosynthetic analysis used in the planning of organic syntheses. The molecule of interest is transformed into precursor molecules by separating it at bonds for which a synthetic reaction is known. In this way, information about chemical reactions is used in a 'backwards'-directed manner, i.e., to deconstruct a molecule into accessible and available building blocks. The underlying hypothesis is that it is more likely to generate new compounds from the building blocks of known molecules. These fragments have been part of an actual molecule and may therefore be synthetically more accessible than molecules designed with other methods.

```
1)
[C;!D4;!D1;!R;!$(C=[C,O,S,N]);$(C([#6;R])!=[#6;!D1]~[!D1]);!$(C[O,S,N;!R]),?R8?]-;!@[C;D3;R;$([C]@[N,n,S,s,O,o]),?R13?]
2)
[C;!D4;!D1;!R;!$(C=[C,O,S,N]);$(C([#6;R])!=[#6;!D1]~[!D1]);!$(C[O,S,N;!R]),?R8?]-;!@[c;X3;R;!$([c]@[N,n,S,s,O,o]),?R16?]
3)
[C;!D4;!D1;!R;!$(C(=S));$(C([S;D2])[#6;!D1]~[!D1]),?R4?]-;!@[S;D2;$(S([C;!D1;!R])[#6;!D1][!D1]);!$(SCS);!$(SC=*);!$(S[P,S]),?R11?]
```

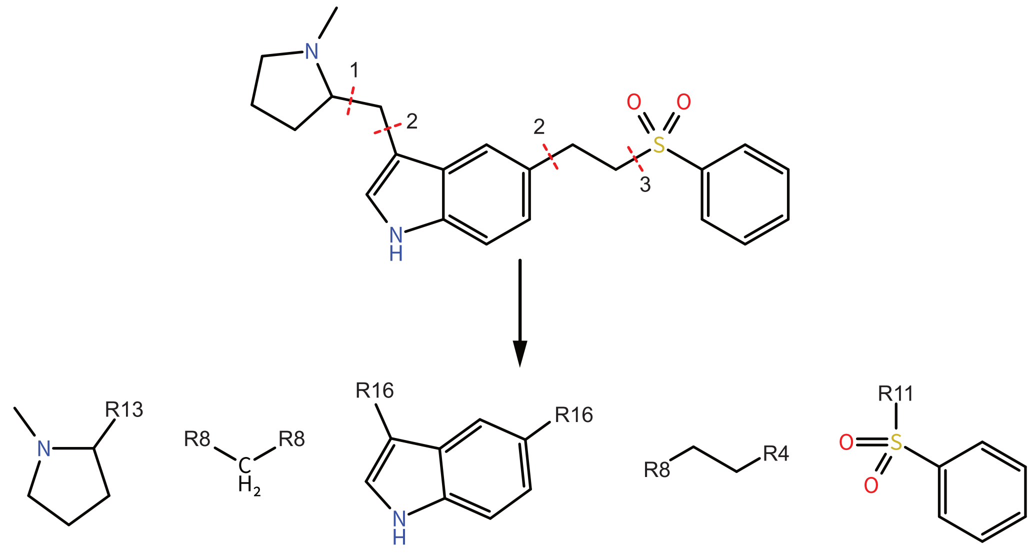


Figure 6.1: Depiction of how a molecule is fragmented with BRICS rules. Top: SMARTS pattern of the three rules that match at least one bond in the molecule. The link types are marked in red. Middle: Molecule Eletriptan (DrugBank DB00216) with the four matched bonds marked by red dashed lines. The number denotes the respective BRICS rule. Bottom: The resulting five fragments after fragmentation.

The selection of initial structures is an important part of the success of this method and should be carried out diligently. The same applies to the selection of suitable fragments. The goal is to avoid large fragments or the introduction of toxic groups to a drug-like chemical space. However, the most important part is to choose the right set of retrosynthetic rules in order to facilitate the generation of molecules in the laboratory. A list with suitable bonds and their environments has to be compiled for the desired reactions. RECAP (retrosynthetic combinatorial analysis procedure) [57] and BRICS (breaking of retrosynthetically interesting chemical substructures) [34] represent such sets (see section 7.2). BRICS attempts to improve the RECAP fragmentation approach by using a more elaborate set of 16 rules and additional post-processing filters. After fragmentation, fragments with more than 16 heavy-atoms, containing rings with more than eight heavy-atoms, or matching a reactive or toxic group according to Kazius et al. [120] are removed. The implementation requires precise information about these bonds as input. Therefore, bonds are defined as SMARTS, i.e., a language for describing molecular patterns based on SMILES [100].

The fragmentation procedures in both cases are very similar. The one for BRICS was originally developed in the Flex* context as part of FlexNovo [25] and is documented in the dissertation of Jörg Degen [26]. Within the scope of this thesis, it was implemented in the NAOMI library. Although the general procedure is straightforward, a few aspects have to be considered when implementing fragmentation of molecules for fragment spaces:

1. In order to be able to recombine fragments efficiently, one must assure that fragments are the smallest possible building blocks. To avoid overlapping fragments, *all* rules are matched to a single molecule at a time in order to identify all cut bonds. Then these cuts are applied exhaustively. If each rule would be applied to an independent instance of a molecules, many redundant fragments would be created.

2. A fragment must occur in a fragment space only once to avoid unnecessary computation when retrieving molecules. The fact that molecules may contain the same substructure can lead to identical fragments. Therefore, the generated candidate fragments need to be filtered for redundant fragments.

The resulting algorithm is depicted in Figure 6.2 and works as follows. First, all valid bonds matching the cut rules are identified for a given molecule using SMARTS patterns [100]. Then, matching bonds are compared with each other to check whether a bond in the molecule has been matched multiple times (see example in Figure 6.2). If this is the case, a bond was found that can be cut with two (or more) retrosynthetic rules. These rules must be applied independently to that bond. Therefore, two (or more) sets of matches are generated, each containing one of the multiple matches. Each resulting set is then translated into cut instructions, which are then applied to separate copies of the molecule. While a bond is removed from the molecule, reaction specific linkers are attached to each adjacent atom[7]. In the case of multiple matches, multiple copies of the fragment with different link types are created. These fragments are added to a set data structure as unique SMILES [114], thus eliminating redundant fragments. After processing all input molecules, fragments are filtered according to BRICS rules. The result is a non-redundant set with little or no overlap between individual fragments.

[7]This is exactly opposite to the process of connecting fragments as depicted in Figure 3.2.

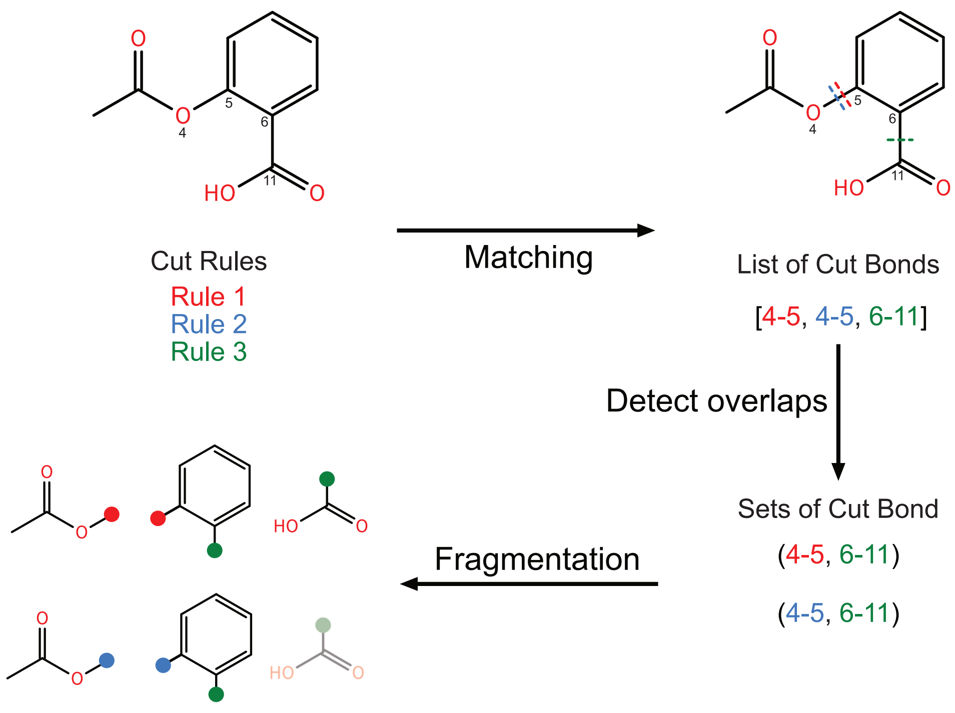


Figure 6.2: Shredding procedure shown for Acetylsalicylic acid (DrugBank DB00945) and fictitious cut rules shown in red, blue, and green. The first processing step matches the cut rules onto the molecule. Note that the list contains bond 4-5 twice because it was matched by different cut rules. Then, multiple matches are detected and two sets with cut bonds are created. Finally, the molecule is fragmented with each set resulting in fragments that are identical except for the link types (color of circles). The faded fragment is the only one that was generated twice.

This functionality is available as standalone command-line tool. For more information see section 9.4.

Retrosynthetic fragmentation can handle large amounts of molecules to quickly generate fragment spaces. Although fragmentation rules are based on reaction information there is some uncertainty inherent to this approach. There is no guarantee that a certain bond between two fragments can in fact be formed with the reaction that the cut rule is based on. In the following section, an alternative approach using reactions in a more direct manner is described.

6.2 Reaction-based Construction

In contrast to the retrosynthetic fragmentation approach, reactions can also be used in a 'forward-directed' way for generating fragment spaces. This corresponds to the actual synthesis carried out in the laboratory, i.e., molecules are mixed in a reaction vessel and thus form a new molecule. The reaction-based approach requires two inputs, a list of molecules and a list of reactions. The former should be a library of small, accessible molecules that are in stock or readily synthesizable, for example an in-house library or molecules that can be ordered from a vendor. The functional groups for each reaction are identified and annotated based on information extracted from the reactions. Then, linkers are added to the molecule at the bond that would be newly formed by this reaction. When two fragments are connected, the linkers are removed and a bond is added (see Figure 3.2).

In the following, the method for generating reaction-based fragment spaces will be described in detail. The algorithm has been newly developed and was implemented as part of the NAOMI library. As a very first version of this new concept, it should be considered a "proof of concept" because there is still room for optimization of the method and adjusting its functionality to specific use cases.

6.2.1 Description of Reactions

The SMIRKS reaction transform language is used to describe reactions. It is based on SMILES and SMARTS [100]. A SMIRKS reaction consists of a list of SMARTS expressions that describe reactants, products and optional agents. These components are separated by greater-than signs (>) thus denoting the direction of the reaction. Chemical entities that have the same role, e.g., several products, are separated by dots (`.`). For example, the SMIRKS for peptide bond formation is:

```
[C;$(CCN):1](=[O:2])O.[N:4][C;$(CC(=O)O):5]>>[C:1](=[O:2])[N:4][C:5]
```

The reaction has two reactants, one product and no agents. The latter would be placed in-between the two greater-than signs.

Two aspects should be mentioned at this point: First, only the simplest case of a reaction with two reagents and one product is considered in the following, which is a generally accepted strategy for making these reactions computationally manageable. Second, graph terminology is used in the following since substructure matching with SMARTS is essentially a sub-graph matching problem.

SMIRKS must follow certain rules in order to be valid. The most important ones are:

1. "The reactant and product sides of the transformation are required to have the same numbers and types of mapped atoms and the atom maps must be pairwise. However, non-mapped atoms may be added or deleted during a transformation" [100].

2. "Atomic expressions may be any valid atomic SMARTS expression for nodes where the bonding (connectivity & bond order) doesn't change. Otherwise, the atomic expressions must be valid SMILES" [100].

This means that every atom involved in a reaction is uniquely numbered and can be unambiguously matched from reactant to product. Also, the description of a reaction can use the versatile SMILES and SMARTS syntax for defining the environment of an atom to precisely describe the structural conditions required for a reaction to take place. An example is shown in Figure 6.3.

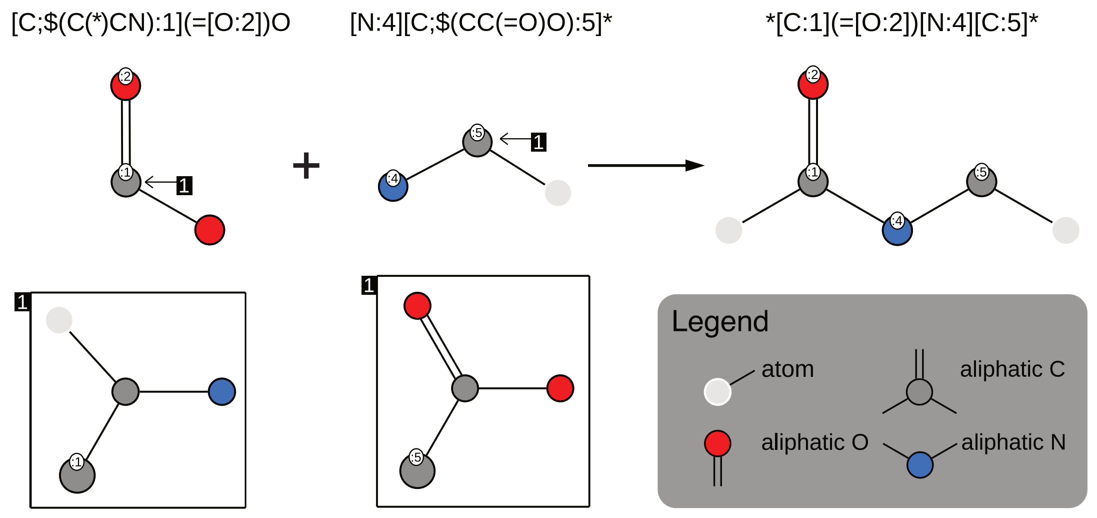


Figure 6.3: Peptide bond formation as described by the SMIRKS `[C;$(CCN):1](=[O:2])O.[N:4][C;$(CC(=O)O):5]>>[C:1](=[O:2])[N:4][C:5]`, depicted as visual SMARTS pattern [62]. The stars (*) in the SMARTS represent the rest of the molecule not considered in the reaction. The fragments in the boxes denoted with "1" describe the environment at the reactive site that is required for (but not changed by) the reaction. The numbers on the atoms are the unique numbering required to map the changes made by the reaction. The reaction introduces a new bond between nodes 1 and 4 thus splitting off a hydroxyl group and a hydrogen.

The implementation uses the SMARTS parsing and matching capabilities of the NAOMI library [62, 121]. In order to determine whether the substructure described by a SMARTS graph is present in a molecule, the nodes of the SMARTS graph are matched onto the atoms of that molecule. This returns a list of matched atoms for each complete match in the molecule.

6.2.2 Processing of Reactions

In the first phase of the algorithm, the SMIRKS expression is parsed and used to identify the involved atoms for each input reaction. The following steps are carried out on the SMARTS graphs of the two reactants and the product, exclusively. First, the node IDs are retrieved from each reactant graph. Then, these IDs are used to find the new edge in the product graph, i.e., the bond that forms during the reaction (see Code Listing 6.1). The two adjacent nodes of the product graph are then marked in the reactant graphs as 'connecting nodes'. These nodes later identify the *connecting atoms* in the molecules that match a reactant. Each reactant is associated with a specific link type which is used in the construction of a fragment. It denotes to which reaction the annotated fragment is compatible and is used to construct connection rules for the fragment space (see section 3.4). The link type must be provided by the user in the same file as the SMIRKS strings. Figure 6.4 shows an example of which atoms are matched and annotated, what the resulting fragments are, and possible molecules assembled from these fragments.

Code listing 6.1 Pseudocode of the function to identify the connecting atoms.

```
FIND-CONNECTING-NODE(reaction)
for each Node $n_1 \in reaction.reactant_1$
    for each Node $n_2 \in reaction.reactant_2$
        if $(n_1, n_2) \in reaction.product$
            MARK_AS_CONNECTING_NODE($n_1$)
            MARK_AS_CONNECTING_NODE($n_2$)
```

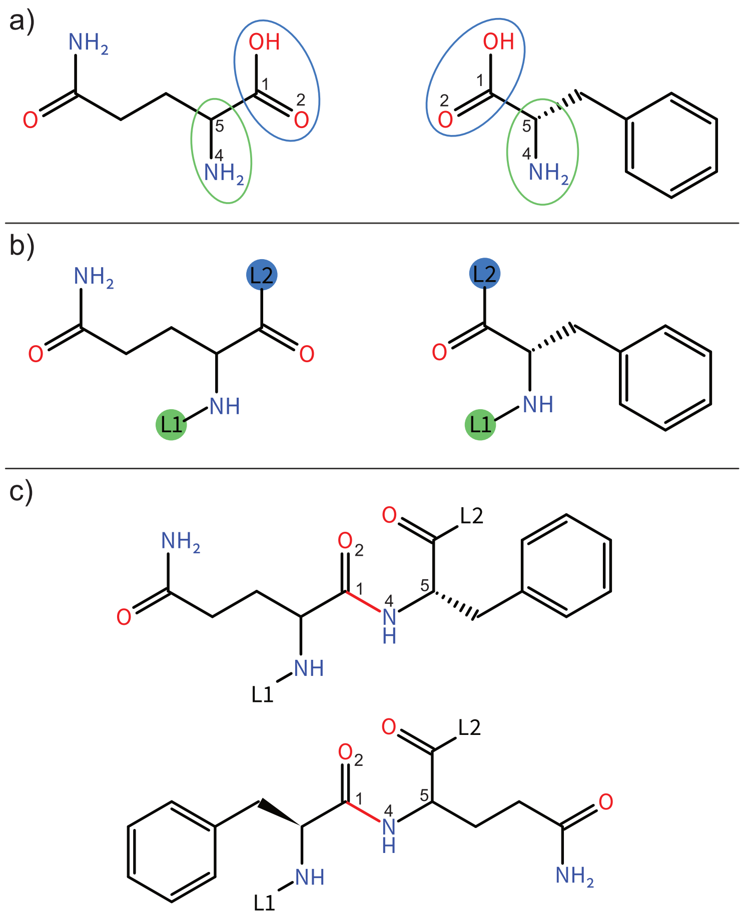


Figure 6.4: Reaction matching for peptide bond forming reaction (Figure 6.3) and two amino acids glutamine (left) and phenylalanine (right). a) Matching of the SMIRKS pattern for reactant 1 shown in blue, and reactant 2 shown in green. Numbers denote the atom ID as defined in the reaction. Atoms 1 and 4 are *connecting atoms* (see c). b) Resulting fragments with annotated linkers. c) Product molecules that may be formed by the two fragments. The newly formed bond is depicted in red.

6.2.3 Matching of Molecules

In the context of reaction-based fragment space construction, the requirement for the matching strategy is that a single fragment should be annotated with reactive sites of different reactions. In this respect, there are two points to consider: First, a functional group may be involved in several reactions and, second, a molecule may have several functional groups.

The first aspect is dealt with by the fragment space model itself. By definition, a given link type can be compatible to several other link types. As a result, a single fragment is stored when multiple identical fragments would be generated that only differ in their link type. This is the case when the same functional group can be involved in the same reaction. This reduces the number of generated fragments significantly and helps to store large amounts of molecules in a compact format. At the moment, the user is required to provide the link types for each reaction. An automated assignment is also possible and could be added as an improvement.

The second part needs more consideration. By annotating a single molecule with a single reaction, the resulting fragments have one linker type only, thus essentially representing a combinatorial library. If this is repeatedly done, a fragment space can represent several but disjoint libraries. Although this is a valid use case for fragment spaces, one can only retrieve molecules built from two fragments connected by a single reaction. To utilize the full potential of fragment spaces, it should be possible to use several reactions to build a new product molecule. In order to facilitate the construction of such fragment spaces, two different strategies were implemented and compared:

(I) The first strategy iterates all molecules in the outer loop and all reactions in the inner loop. For each molecule, it matches all reactions at once, thus generating a potentially large number of matches. In order to apply these matches, it determines which matches overlap. From this information, non-overlapping *match sets* are generated and applied to a copy of the molecule independently, each resulting in a distinct fragment. By matching all reactions at once, the reactive sites for different reactions can be annotated at once.

(II) The second strategy iterates all reactions in the outer loop and all molecules in the inner loop (see Code listing 6.2). For each reaction, the algorithm iterates all molecules and, in addition, all fragments from previous iterations. It matches a single reaction to a single molecule (or fragment) and annotates the reactive sites of this reaction only. In this manner, functional groups are annotated in an iterative fashion rather than at once. By matching existing fragments, reactive sites belonging to different reactions are annotated as well.

Code listing 6.2 Pseudocode of the general strategy of the reaction-based construction of fragment spaces.

```
RXN-FS(molecules, reactions)
fragments = Set.init()
for each Reaction $r \in$ reactions
    for each Fragment $f \in$ fragments
        MATCH-AND-ANNOTATE($r$, $f$, fragments)
    for each Molecule $m \in$ molecules
        MATCH-AND-ANNOTATE($r$, $m$, fragments)
return fragments
```

In either case, the following procedure is used to determine if a reaction can be applied to a molecule. Please refer to Code listing 6.3 for an overview. Both reactant graphs of a reaction are matched onto a single molecule independently, which is necessary because molecules may have several functional groups and serve as either reactant[8]. The matches from both reactants are combined. There are occasions were reactant SMARTS match the same atoms so that matches overlap and the annotation of both reactive groups is not possible (see Figure 6.5). This is similar to the overlapping bond matches occurring in the fragmentation algorithm (see section 6.1). Here, this is dealt with by assigning unique *match IDs* to each match. After all matches have been identified, they are sorted into sets that only contain the IDs of non-overlapping matches. Once all distinct match sets are created, each is translated into a cut instruction. A cut instruction describes which atom to remove from the *connecting atom* identified during the initial processing of the reaction. This atom can either be an implicit hydrogen or an unlabeled heavy atom. The latter is described by rule 1 of the SMIRKS definition (see 6.2.1); an example for this is the single-bonded oxygen of the first reactant in Figure 6.3. The identified atom is subsequently replaced by the reaction-specific linker. The resulting sets of cut instruction are applied to separate copies of the molecule, thus generating all possible permutations of linkers.

Any reaction may introduce bond changes in the molecule that lead to changes in the valence state of individual atoms. The resulting fragment must exhibit the bond and valence state as present in the product when it is entered into a

[8]A prominent example for this is amino acids (see Figure 6.4).

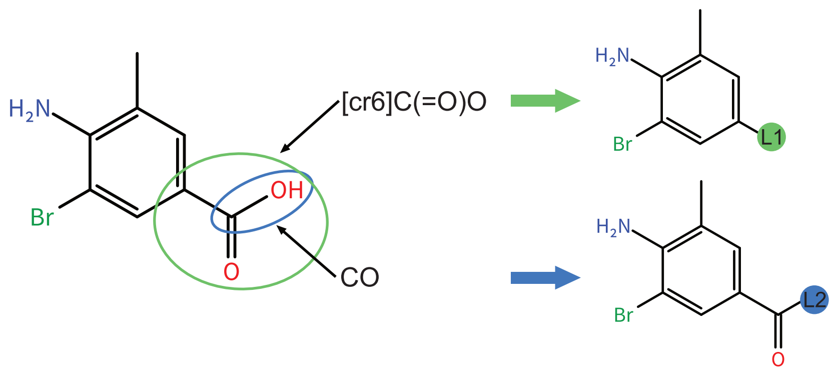


Figure 6.5: Example of overlapping reactive groups. The matching atoms for two artificial reactant SMARTS are encircled in the molecule. The first SMARTS matches a carboxylic acid at a aromatic ring, the second matches an alcohol. The resulting fragments for both cases are shown on the right. The molecule is part of the ChemBridge Building Blocks dataset (ZINC4050446).

fragment space so that no modifications are required when it is later retrieved from a fragment space. Currently, only bond changes at the *connecting atom* and the atom that is replaced with a linker are considered, i.e., changes that can be compensated by adding or splitting off a hydrogen at the *connecting atom*. For example, in the case of reductive amination (see Figure 6.6), a double bonded oxygen is removed and a single bonded linker is added because the product only contains a single bond. This results in an open valence that is saturated by adding a hydrogen to the reactive atom. More complicated bond changes are not modeled at the moment, as further discussed in section 8.3.

Determining Non-overlapping Matches

Determining all non-overlapping match sets is a relatively expensive computation. The algorithm computes the pairwise overlap of all matches and stores this information in the overlap matrix. This matrix is symmetrical and forms the basis for the following computation. For each row, the algorithm carries out the following steps: First, it adds all match IDs as initial members of a match set. This set is added to a queue, which is subsequently processed until it is empty. A match set from the queue is iterated via two nested loops to carry out a pairwise comparison of all members. If two members are found to overlap based on the compatibility matrix, two sets are created from the initial set: Each set contains one of the incompatible match IDs and all previously added match IDs. These sets are added to the queue. If no overlap is found, the set only contains non-overlapping matches and becomes part of the result list. This list may contain subsets of larger match sets, which are filtered out subsequently.

Code listing 6.3 Pseudocode of the matching process. First, all possible matches are searched and combined, then, they are applied.

```
MATCH-AND-ANNOTATE($reaction, molecule, fragments$)
$matches1$ = List.init()
$matches2$ = List.init()
$disjoint\_matches$ = List.init()

SMARTS-MATCHER($molecule$, $reaction$.reactant_smarts1, $matches1$)
SMARTS-MATCHER($molecule$, $reaction$.reactant_smarts2, $matches2$)
COMBINE-LIST-OF-MATCHES($matches1, matches2, disjoint\_matches$)

for each MatchSet $match\_set \in disjoint\_matches$
    $split\_bonds$ = List.init()
    for each Match $m \in match\_set$
        $cut\_bond$ = Bond.init($m$.connecting_atom, $m$.split_atom)
        $split\_bonds$.insert(cut_bond)
    SPLIT-MOLECULE-AND-ADD-LINKER($m, split\_bonds, fragments$)
```

The current implementation is not very efficient and could be optimized in future versions. It requires frequent lookups in the overlap matrix and uses several nested loops. The more matches are involved, the larger the compatibility matrix and the more complex the computation of permutations. This turned out to be problematic for strategy I discussed above. When applying all reactions at once, some molecules lead to an impractically long computing time. As a consequence, strategy I was discarded in favor of strategy II. During the latter, much fewer matches per molecule occur since only one reaction is matched at a time. Consequently, strategy II was implemented into a command-line tool. For more information, see 9.3.

6.2.4 Conclusion

In the sections above, a second approach to generate fragment spaces based on reaction information was introduced. Utilizing reactions in a forward-directed way has two major advantages over retrosynthetic fragmentation. One is the use of non-theoretical building blocks in the form of synthetically available molecules and the other advantage is the use of organic synthesis reactions in a direct, non-abstract way. These two features are the basis for an easy transition into the laboratory. For each product molecule, a synthesis protocol can be derived based

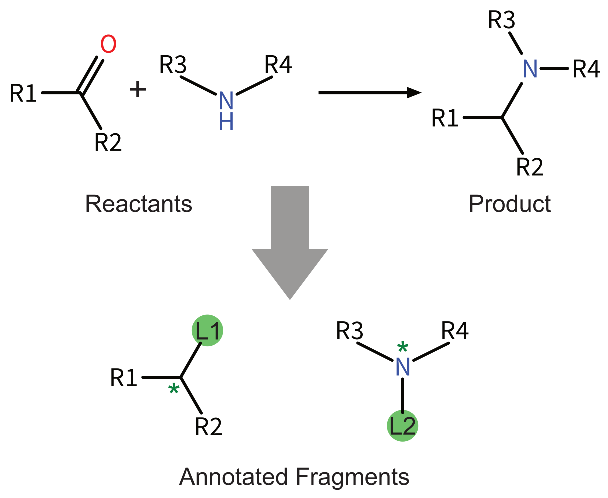


Figure 6.6: Reductive amination as an example for a reaction with a bond change. The information to annotate fragments is extracted from the reaction definition as shown on top. The resulting fragments are shown on the bottom. The connecting atoms are marked with an asterisk, the annotated linkers are shaded in green. In this case, R1-R4 denote those parts that differ between molecules and are not relevant for the reaction.

on the building blocks and the connection rules involved. In chapter 8, experiments for the construction of fragment spaces with this method are presented. Improvements for this proof-of-concept implementation are discussed in chapter 10.

6.3 Existing Enumeration Algorithm

The previous implementation of the fragment space enumeration algorithm by Pärn, Degen, and Rarey is based on the Flex* library [29]. The user must provide a fragment space and physicochemical constraints as min-max range. *FlexEnum* uses a tree data structure to represent molecules during enumeration. The generated trees are systematically extended by adding one fragment at a time and confirming the validity of the properties of the tree. This is done recursively. In the corresponding publication, three problems are defined: "how to access fragments with certain properties in an efficient way, how to neglect redundant fragment-trees as soon as possible and how to confirm the uniqueness of a generated molecule effectively" [29].

The first problem is addressed by utilizing a kD-tree as an index for fragments. This tree can be queried with property ranges and returns a list of fragments that do not violate the boundary constraints. The problem of redundant trees is dealt with by defining rules for the attachment of fragments. Since these rules are also part of the new implementation that was developed during this project, they are discussed in detail in section 6.4.5. In order to determine the uniqueness of a molecule, a self-balancing binary search tree is used to store a string representation of the corresponding molecule. The unique SMILES [114] line notation is used for this.

In the accompanying study, 40 fragment spaces were generated with modified RECAP rules and enumerated. The spaces contained 4 to 271 fragments and yielded between 21 and 779,213 molecules. Nine enumerations did not finish as "a consequence of the limited memory on the Linux nodes [...] and also due to the size of the fragment spaces." In addition, an experiment to demonstrate the performance of the redundancy filter was discussed. Therefore, two artificial fragment spaces with 47 and 48 fragments were used, yielding 73,147 and 95,713 molecules. This space was also enumerated with the newly developed algorithm using the same constraints (see B.4).

In the following, the new enumeration algorithm is described. It is related to this approach and uses the same overall strategy. Molecules are extended one fragment at a time and are kept as a tree data structure on which redundancy prevention strategies can be applied. Then, newly generated molecules are checked for redundancy. However, most of the implementation was done in a different and improved way. For instance, instead of a recursive strategy an iterative one was used, the redundancy prevention strategy was expanded, and redundancy checking was done in a memory-independent way. The differences of both methods are highlighted throughout section 6.4.

6.4 Enumeration

In the following, the Fragment Space exhaustive enumeration system (FSees) is described[9] [1]. The goal of this new enumeration algorithm is to find all molecules within a chemical subspace that is defined by the following constraints:

1. the structure of the fragments in the fragment space,
2. the synthetic rules encoded in the connection rules, and
3. the physicochemical properties that the user has entered.

In order to accomplish this, the enumeration procedure works as follows. It uses every fragment in the fragment space as a starting point. For each fragment,

[9]The status of the accompanying publication “FSees: Customized Enumeration of Chemical Subspaces with Limited Main Memory Consumption” [1] is "2016".

it exhaustively explores all possibilities for extension within the given physicochemical constraints. For each open link, all fragments with compatible linkers are attached to a copy of the initial fragment. These new, conjugated fragments are then subject to the same attachment strategy until no further extension is possible. This process is deterministic and exhaustive. The extension of an individual molecule is stopped only when its physicochemical properties are outside the user-defined range, if the user-defined maximal number of fragments is reached, or if it is identical to a previously encountered molecule.

In order to be able to enumerate a large chemical space with millions of molecules, several strategies to save computer resources were implemented. First, the algorithm is designed to operate with constant main memory consumption. Therefore, it uses file-based data structures thus rendering the disk space the limiting factor rather than the main memory (see section 6.4.4 and 6.4.6). Second, unnecessary computation is prevented by avoiding redundant molecules. This leads to a reduction of runtime and overall resource requirements. As discussed in section 6.4.5 and 6.4.6, redundancy cannot be completely avoided, but it can be detected. Third, the calculation of physicochemical properties is done efficiently (see section 6.4.2 and 6.4.3). This is combined with the detection of redundant structures so that expensive calculations are carried out only after all other filters were passed (see section 6.4.7).

6.4.1 Topology

In order to calculate properties, it would suffice to keep all involved fragments in a simple list. However, structural information would not be stored. When connecting several fragments, it is important to preserve the information about which linker of fragment A is used to connect to fragment B and vice versa. This information determines the topology of a molecule.

The straightforward way to preserve the topology is to simply connect the fragments thus creating a new molecule. However, connecting fragments is an expensive operation since all atoms, bonds, and associated data of the underlying molecule instance must be copied. Due to their properties, many molecules will be sorted out anyway. Executing an expensive operation to immediately discard the resulting molecule is a waste of resources. In addition, it is not required since many properties can be calculated on individual fragments (see section 6.4.2). Hence, fragments are connected only when absolutely necessary. Until then, a list of fragments, which are meant to be connected into a molecule, needs to be stored in a way that allows for it to be processed as a molecule with respect to its topology. A tree data structure is utilized for this purpose (see 6.4.3).

6.4.2 Property Computation

The FSees algorithm systematically combines fragments in order to assemble new molecules. Each new molecule must be evaluated with respect to its physicochemical properties. However, combining fragments into a molecule is an expensive operation. In order to calculate the properties of a molecule, this can be avoided in certain cases. The properties considered in this algorithm can be described by a single numerical value, which is either an integer or a real number. They are computed by NAOMI library functions [59] based on the molecule data structure. Since a fragment is simply a small molecule (see section 3.4), these functions can calculate the respective properties for a fragment only and hence it is not necessary to connect fragments in order to calculate certain properties for a molecule.

There are two categories of properties, i.e., those that are monotonously additive and those that are not. This differentiation was also made by Pärn et al. in the previous implementation [29].

Definition 6.1. A property X is named **monotonously additive**, if X can be calculated for a molecule by adding up the individual values of X for each fragment of which the molecule consists and the sign of X is constantly either non-positive or non-negative. [10]

Many of the physicochemical properties fulfill the criterion for *monotonously additive* and can therefore be calculated for each fragment individually. Consequently, it suffices to sum up the individual values of a list of fragments in order to calculate the value for a molecule. Examples are the number of hydrogen bond donors, the molecular mass, or – less obviously – the calculated volume (tVolume). The latter is an approximation of the volume of a molecule not taking an overlap between atoms into account [58]. For a complete list of properties, please refer to Table 6.1.

Linkers are treated as atoms without properties. More precisely, the values for their atomic properties, e.g., volume or mass, are zero. For the computation of molecular properties they are not considered by the particular library function.

The available non-monotonously properties are clog$P$ (calculated according to Wildman et al. [122]), rotatable bonds, stereo-centers, and topological polar surface area (tPSA). The algorithms for computing these properties take neighboring atoms with varying depth into account. Therefore, the contribution of a single fragment is dependent on neighboring fragments. In these cases, a new molecule is actually created from these fragments in order to compute these properties.

An alternative approach could be the adaption of each algorithm to work with unconnected fragments. When encountering a linker, the algorithm would have

[10]This definition was first used in [1]

to continue processing the fragment adjacent to this position. This could, in fact, be realized based on the fragment tree data structure that will be introduced in the next section. However, this was not done because it would require a new algorithm every time a non-monotonously additive property was added to the FSees enumeration algorithm. Using native NAOMI functionality allows for convenient extension of the method with additional properties. In order to limit the effect of the expensive construction of a molecule on runtime, this operation is only carried out when all other filters were passed (see section 6.4.7).

Table 6.1: List of properties that can be used for enumeration and their category. *Other* refers to the fact that these are not molecular properties.

Property	Monotonously additive	Non-monotonously additive	Other
Atoms	×		
clogP		×	
Fragments			×
H-bond acceptors	×		
H-bond donors	×		
Lipinski-style acceptors	×		
Lipinski-style donors	×		
Molecular weight	×		
Non-hydrogen atoms	×		
Non-hydrogen bonds	×		
Rings	×		
Rotatable bonds		×	
Stereo-centers		×	
tPSA		×	
tVolume [58]	×		

6.4.3 Fragment Tree

The *fragment tree* data structure stores how fragments are interconnected, it is used to calculate molecular properties efficiently, and serves to identify redundant structures. It is the central data structure of the FSees algorithm. A node of the fragment tree represents a specific fragment and an edge represents a bond between these fragments. Hence, a fragment tree constitutes a topological representation of the molecule.

A node contains a pointer to the fragment instance so that fragments exist in memory only once. It has as many edges as the corresponding fragment has linkers. The edges on the node are sorted according to the atom IDs of the corresponding linkers on the fragment, i.e., a molecule instance. This sorting is important for the unambiguous assignment of linkers regarding the construction of a descriptor that is used to uniquely identify a tree. Edges are modeled implicitly, i.e., as pointers to other nodes. Hence, a node contains a list of pointers to its child nodes and one to its parent node. The root node represents a special case; it only has child nodes. When a node is added to a tree, i.e., when two fragments are connected, these pointers are set accordingly: The child pointer of the parent node is pointed at the new node and the parent pointer of the new node is pointed at the parent node.

Fragment trees are iteratively extended during the enumeration process one fragment and node, at a time. Therefore a fragment may still have several open linkers after it has been added to a tree. To ensure that the resulting molecule does not have open linkers, a *terminal fragment* is added to the tree for each unconnected edge. The saturation of open linkers is necessary since link atoms do not represent chemically valid atoms, but rather open valences. This is particularly important for the computation of properties. Terminal fragments are provided by the corresponding fragment space for each link type.

Since fragment trees represent a precise description of a molecule, they are used to detect redundancy. Therefore, a means to compare fragment trees is required and for this purpose, a linear string representation is generated by a DFS-based algorithm. This string representation uses a human readable format that was used for debugging. A node is uniquely identified via its fragment ID, followed by a list of child nodes. The node ID is followed by a colon and the list of children is put in square brackets. In this list, the parent node and a terminal fragment are represented by the letter P and T, respectively. Since children are sorted according to their atom ID as described earlier, this representation is unambiguous. An example is shown in Figure 6.7.

6.4.4 Enumeration of Fragment Trees

The FSees algorithm is designed to exhaustively explore the chemical space of a fragment space. The overall strategy was outlined at the beginning of section 6.4. The core of the algorithm is the extension protocol, which was developed in conjunction with the fragment tree in order to efficiently create new molecules. In the following, a detailed description of the enumeration algorithm is provided. First, the queue data structure is introduced, which is key to this algorithm.

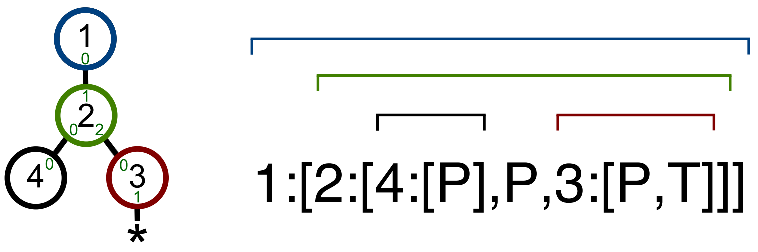


Figure 6.7: Depiction of a fragment tree (left) and the corresponding linear descriptor (right): large numbers denote the node ID, small green numbers the edge ID, and an asterisk the unsaturated linker. The edge ID has no significance outside the node; it is used internally to map a certain edge to the same linker on the fragment in every instance.

Fragment Tree Queue

In the version by Pärn et al., a recursive approach was used to extend a fragment tree [29]. However, this poses a problem regarding memory consumption. For every open linker and every compatible fragment, a copy of the current fragment tree is needed, to which the compatible fragment can be attached. In the recursive case, these copies all need to be kept in memory. This is not a sensible strategy especially since the goal is to be independent of main memory. The algorithm developed for this project, FSees (Fragment Space exhaustive enumeration system), uses an iterative approach based on a queue.

To avoid the exhaustive use of memory, a custom, file-based queue data structure was implemented to store fragment trees for further processing. This implementation uses the SQLite library, which "implements a self-contained, serverless, zero-configuration, transactional SQL database engine" [123]. SQLite was chosen because it is file-based and provides a convenient API through the Qt application framework[11], which is employed by the NAOMI library. During the development of this algorithm, it was first used for redundancy testing, which is described in section 6.4.6.

The database contains a single table with a single column storing serialized fragment trees. This serialized version is different from the linear descriptor (see 6.4.3). Its purpose is to efficiently serialize and deserialize a fragment tree without the requirement for easy comparability. The fragment tree is converted to a byte array and stored as a BLOB in the database. For efficiency, the queue data structure uses a buffer, which is a memory-based queue that stores fragment tree instances. The buffer queue is allowed to contain a certain amount of elements s_b. If this amount is exceeded by the chunk size s_c, the last s_c added molecules

[11]http://www.qt.io/

are serialized and written to the database queue. If the buffer queue is empty, the database queue is queried for the first s_c elements which are then deserialized and added to the buffer queue.

At the beginning of the enumeration, a single instance of this queue, i.e., the *tree queue* is initialized. The next section explains how it is utilized.

Extension Protocol

The extension protocol represents the central logic responsible for the construction of molecules. It utilizes an iterative procedure based on the tree queue rather than a recursive scheme. A depiction of the algorithm is shown in Figure 6.8.

To exhaustively enumerate a given fragment space, all fragments are used as starting points. The fragment space is iterated to prepare fragments for processing. For each starting fragment, a fragment tree is constructed with this fragment as root. The resulting trees are processed individually. An initial tree is only processed after the previous one has been completely explored.

The extension protocol is triggered when an initial tree is added to the *tree queue* (see Figure 6.8 black arrow). First, only one tree is present in the queue. While the algorithm explores all possibilities for extension, new trees are added constantly. Only when the queue is empty, the next fragment tree corresponding to a starting fragment is added and the extension protocol is carried out again. The algorithm terminates when every starting fragment has been processed by the extension protocol.

The extension protocol consists of three nested loops (see Figure 6.8):

Loop 1 accesses the tree queue by dequeuing fragment trees until it is empty, i.e., until all trees have been processed. For each tree, it then retrieves all open linkers.

Loop 2 iterates open linkers of the *current tree* and determines all compatible fragments. This is done efficiently by querying the fragment space index provided by the NAOMI library (see section 3.4.1).

Loop 3 iterates these compatible fragments. Each fragment is attached to a copy of the current tree as a new node. In this way, a new fragment tree instance is created for each fragment. At this point, a simple yet efficient strategy to prevent unnecessary computation is implemented. The fragments in the fragment space are sorted according to their mass. As soon as the first fragment is attached, which violates a potential upper bound criterion, this loop is aborted. In cases where no such criterion is defined, this abort criterion does not take effect. The new tree is then subject to property filters and redundancy tests. The different steps of these filters are described in detail in the section 6.4.7. During these tests, valid fragment trees are translated into molecules. If the tree and the corresponding molecule pass all tests, the latter is written to the output file. The tree, in turn, is added to the tree queue to become subject to further rounds of attachment until no more fragments can be attached.

In the following, the types of redundancy that can occur on tree level will be discussed. Efficient strategies for avoiding redundancy are introduced accordingly. In conjunction with the redundancy tests, a strategy for preventing the exhaustive use of main memory was developed and will be discussed subsequently. During this discussion, the different filter steps in loop 3 will be described in detail.

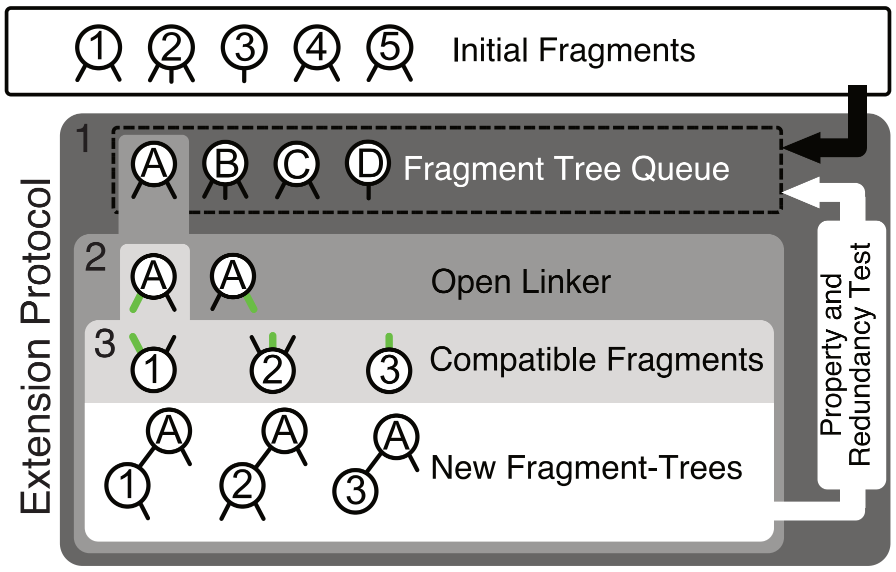


Figure 6.8: Overview of the Enumeration Algorithm. Solid boxes show the three nested loops. A circle with a number represents a fragment, while a circle with a letter represents a fragment tree from a previous iteration. A green line depicts open and compatible linkers. initial fragment trees are added to the tree queue via the black arrow, new extended fragment trees via the white arrow. *This figure was originally published in [1].*

6.4.5 Avoiding Redundancy

When enumerating fragment trees, several configurations would lead to the same molecule. They can be prevented when constructing the fragment tree before a compatible fragment is attached to a tree (see section 6.4.4). Some of these cases can be prevented by simple tree-topological considerations (see Figure 6.9 I-III). Cases I and II have been previously discussed by Pärn et al. [29] and use the fragment ID. Case III prevents redundancy that stems from symmetric fragments and was first developed within the scope of this project.

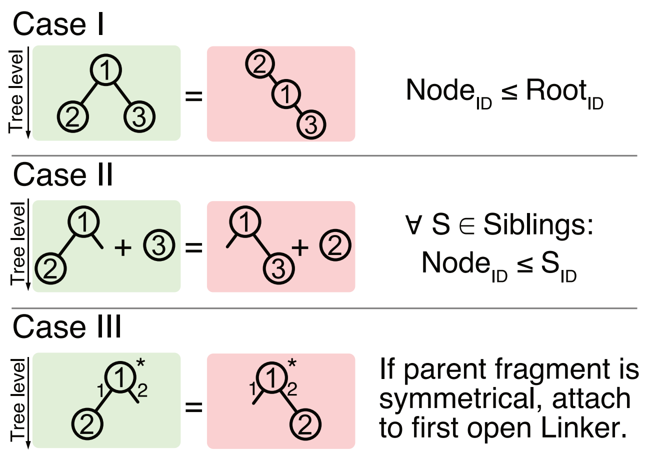

Case I
Tree level
=
Node_ID ≤ Root_ID
Case II
+
=
∀ S ∈ Siblings: Node_ID ≤ S_ID
Case III
If parent fragment is symmetrical, attach to first open Linker.

Figure 6.9: Overview of the resolved types of redundancy. I-III: On the left fragment trees are shown. The green background depicts the fragment tree that occurs first, red the identical tree that occurs later during enumeration. The latter are prevented by the rule on the right. * marks symmetric fragments. *This figure was originally published in [1].*

Fragment ID

Assuming a fragment space contains fragments 1, 2, and 3 as depicted in Figure 6.9.

Case I: When fragment 1 is the starting fragment, a resulting tree could be the one in the green box. Later, when fragment 2 becomes the starting fragment, the tree in the red box may be created. Assuming fragment 2 and 3 are connected to the same linker of fragment 2 in both cases, both trees represent the exact same molecule. A simple order rule developed by Pärn et al. can prevent the occurrence of such redundant trees [29], reducing the number of redundant trees substantially in an efficient way: A fragment can only be attached, if its ID is greater than or equal to the ID of the root fragment.

Case II: If fragment 1 is the starting fragment, attaching fragment 2 first and 3 later leads to the same molecule as attaching fragment 3 first and 2 later. This case can be prevented too, with a simple order rule by Pärn et al. [29]: A fragment can only be attached, if its ID is greater than or equal to the ID of any already existing sibling fragment.

These two simple rules prevent identical trees to a large degree. However, they do not consider symmetric fragments, which also cause redundant trees to occur.

Symmetry

Symmetry in this case refers to topological symmetry in a fragment concerning the arrangement of linkers. Consider **Case III** in Figure 6.9. If fragment 1 is symmetrical, attaching a fragment to either linker leads to the same molecule. This causes the generation of many redundant molecules every time a symmetric fragment is encountered. The solution is to use a symmetric linker position only once per iteration. This ensures that a specific molecule is created only once and therefore becomes subject to extension only once. The rule that can be derived for symmetric fragments is: *Attach an additional fragment only to the* ***first*** *open linker per iteration.* The exclusion of all but one linker position assures that during each iteration, only the first of the remaining open linkers is considered. In this way, symmetric linkers are saturated in consecutive iterations rather than in the same one. Figure 6.10 shows an example of the generated fragment tree with and without the additional rule. Without this rule, a large number of redundant structures is generated, which would have to be sorted out later. With the new rule, each tree is created only once instead.

In oder to consider symmetry during enumeration, symmetric fragments have to be identified first. Therefore, the automorphism generator from the NAOMI library was used. For a given molecule, it generates all permutations of atom IDs that represent automorphisms and then it checks if the atom IDs of the linkers are at the same position in each permutation. If they are not, they are symmetric. The sorted list of symmetric linkers is stored in the fragment tree node. Every time the algorithm encounters a symmetric fragment, the compatible fragment is attached to the first open linker.

Although a lot of redundant fragment trees can be prevented by adding this simple rule, this approach works only for fragments with two and three symmetrically placed linkers. The resolution for fragments with four and more symmetric linkers – possibly of different link types – is cumbersome for combinatorial reasons (see Figure 6.11 a). Since fragments with four and more linkers are rare, they are not taken into consideration. Another type of symmetry called *implied symmetry* occurs when a fragment has linkers with different types but the same compatibility so that identical fragments can be attached (see Figure 6.11 b). Preventing this kind of symmetric fragment by tree-topological considerations alone is not possible since knowledge about the fragment space needs to be taken into account. Therefore, this source of redundancy is not considered either at this point in the algorithm.

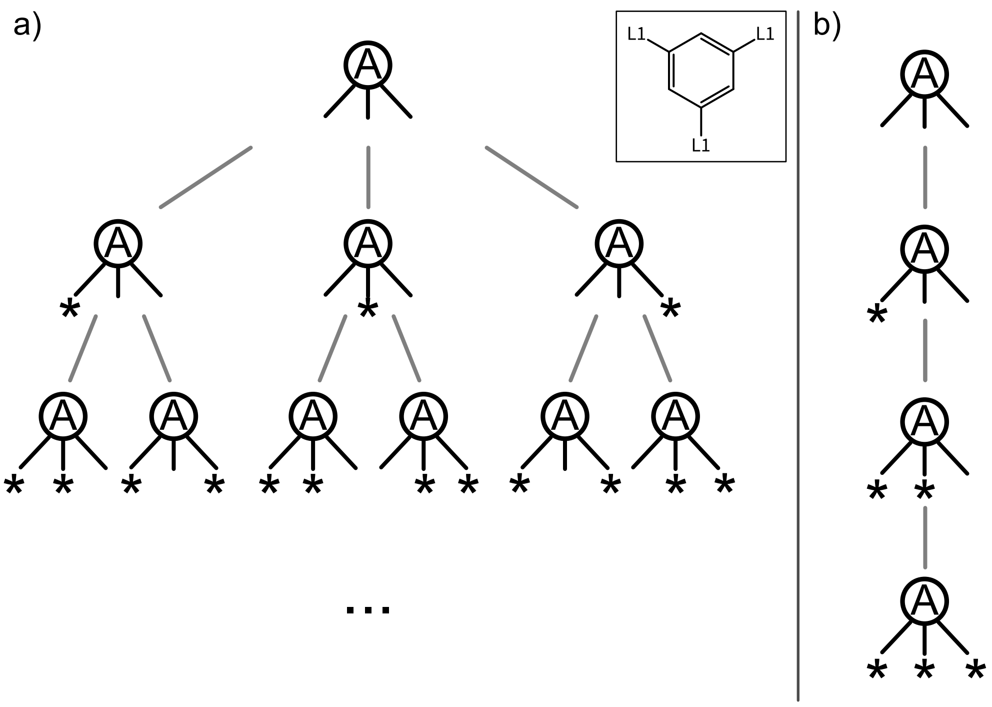


Figure 6.10: Fragment trees generated in each iteration for a symmetric fragment a) without symmetry rule, and b) with symmetry rule. * marks the linker position where a fragment has been attached. Since fragment A is symmetric, all trees on the same level represent the same molecule. An example for a symmetric fragment is shown in the box.

Other Types of Redundancy

By utilizing the previously introduced rules, many redundant structures are avoided. The remaining cases of redundancy cannot be easily resolved based on the fragment tree topology (see Figure 6.11 a-d). The first three cases are directly related to symmetry: (a) and (b) were discussed in section 6.4.5. In case (c), the connection of two or more fragments results in a symmetric substructure, which leads to the same problem as a symmetric fragment. The last case (d) occurs when two or more fragments are connected and the result is identical to another fragment or to several other connected fragments. These cases either require complicated rules for each special case (a), or additional information from the fragment space (b), or can only be detected by a comparison of substructures or molecules (c, d). The simplest solution would be to generate all molecules and apply a subsequent filtering step to remove duplicates. However, it is better to detect redundancy as early as possible to avoid unnecessary computation due to structures that have already been encountered.

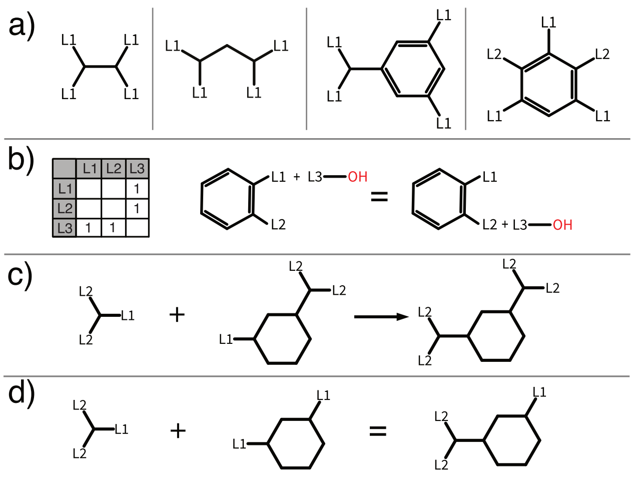


Figure 6.11: Overview of the types of redundancy that cannot be detected based on fragment tree topoplogy. a) Examples for symmetric fragments with more than three linkers. b) Fragment with different link types that is symmetric due to the compatibility rules depicted in the compatibility matrix. c) Example for two fragments that results in a symmetric product. d) Example for two fragments that, upon connection, have the same substructure as a third fragment. *This figure was originally published in [1].*

6.4.6 Detecting redundancy

In the previous section, the different strategies for preventing redundancy were discussed. Yet, there are still cases when it is not possible or feasible to avoid redundancy during the construction of fragment trees and molecules. However, repeated exploration of search space is not the best scenario either, because it results in a significant amount of unnecessary computation. Therefore, the detection of redundancy is a sensible strategy.

In order to reliably check for redundant structures, the FSees algorithm keeps track of all fragment tree and molecule instances that occur during the enumeration. Pärn et al. used a similar strategy base on a binary search tree that was kept in memory. However, since a very large number of fragment trees and molecules may be encountered, an efficient storage with requirements for little main memory is needed. A SQLite database [123] is used here as well because it fulfills all requirements: it is file-based, it can hold large numbers of entries, it provides very efficient duplicate checking, and provides an easy to use API, so that an own implementation is not required.

Two database instances are used, the *tree database* and the *molecule database* containing fragment trees and molecules, respectively. Each database contains a single table with a single column to hold a string representation of either fragment tree or molecule. The string representation of the fragment tree is described in section 6.4.3 (see also Figure 6.7). For molecules, a NAOMI specific linear descriptor is used, yet any unique linear descriptor will suffice, e.g., unique SMILES [114] or InChi [124].

Two strategies are applied to make database access more efficient. First, a Bloom-filter [125] is utilized. This is a probabilistic data structure that allows to efficiently check whether an element is not present in a storage yet. It consists of a bit array and a set of k hashing functions. At the beginning, all bits are set to 0. Upon addition of a new element, k hash values are calculated and the corresponding k bits are set to 1. A lookup operation also calculates k hashes for the input and checks whether the corresponding bits are set. If this is not the case – even if only one bit is not set – the element is not present in the database. If all bits are set, the element either exists or it is a false positive match. In this case, the underlying database must be queried in order to rule out a false positive. Since elements cannot be removed from a Bloom filter, the string storage data structure only supports the addition of elements. This suits this use case well since all elements that have ever occurred should be stored. The number of hashing functions is hardcoded; four different functions are used. The size of the Bloom-filter is a parameter, the user can set when starting the enumeration. The ideal size m of the Bloom-filter can be estimated based on the number of expected elements n, the number of hashing functions k, and the desired false positive probability p:

$$m = -\frac{n \ln p}{(\ln 2)^2}$$

If the size of the Bloom-filter is underestimated, the enumeration will still finish successfully. In this case, all queries to the Bloom-filter will simply result in a query to the database.

The second strategy is a buffer for writing operations. By writing several elements to the database at a time, the costly operation of rebuilding the database index does not have to be carried out for every element. The effect of this parameter is discussed in the results section 8.1.

6.4.7 Prioritization of Filters

The procedure for detecting redundancy is carried out within loop 3 after a new fragment tree has been assembled. It is combined with the computation of properties of a fragment tree and the corresponding molecule for efficiency. A depiction of the different filtering steps is shown in Figure 6.12. The most efficient order of filters was empirically tested. First, monotonously additive properties are calculated based on the fragment tree. This step is the computationally least expensive since it merely requires the iteration over fragments of a fragment tree and the summation of values (see 6.4.2). Second, it is checked whether the tree is redundant: The tree is added to the tree database and the Bloom-filter is updated accordingly. The next filters are based on the NAOMI molecule data structure an for that purpose atoms, bonds, and associated data of each fragment are copied and connected to create a new molecule instance. Third, it is checked whether the molecule has been encountered before. A string representation is generated and added to the molecule database. Finally, the non-monotonously additive properties are calculated.

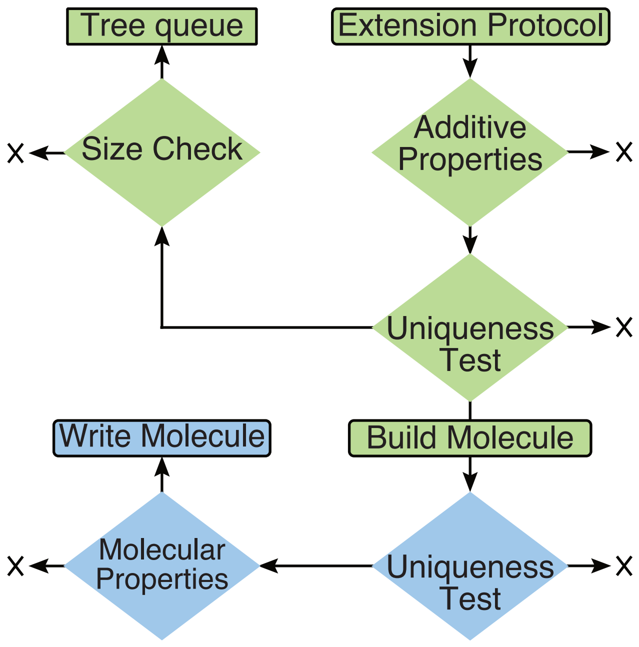


Figure 6.12: Flowchart of the property and redundancy tests carried out in loop 3. The part of the algorithm that works on the fragment tree is shaded in green, the part that works on the molecule data structure in blue. X denotes the rejection of a tree or a molecule. *This figure was originally published in [1].*

This order has proven to be the fastest in the scenarios presented in section 8.2.3. The reason is the costly computation of clog$P$ (see section 8.1.1). Although the computation of TPSA and rotatable bonds is very fast, the calculation of clog$P$ is not. This fact combined with its frequent use in many constraint sets, e.g., lead-like [12, 14] and drug-like [15], makes it the bottleneck of non-monotonously property computation. In the following, the different filters and how they interact are discussed in detail.

The computation of monotonously additive properties is carried out after a new fragment has been attached to the tree. At this point, the calculated values (e.g. number of hydrogen bond donors or acceptors) are compared to user-defined lower and upper thresholds with respect to an optional tolerance value k. The latter describes how many properties are allowed to be out of range while the molecule is still considered valid. There are three possibilities.

1. If more than a number k of properties is above the upper threshold, the tree is discarded since it does not represent a valid molecule.

2. If less than k property values are above the upper threshold, the tree represents a valid molecule. It is checked for redundancy by means of the tree database. If it is new, its linear descriptor is added to the tree database and its fragment tree to the tree queue.

3. If more than k property values are below the lower threshold, the tree does not represent a valid molecule. However, up to this point, only monotonously additive properties were considered. Therefore, adding more fragments may result in a valid molecule. In this case, the tree is added to the tree queue for further extension and the tree database to prevent redundant computation.

A fragment tree is only added to the queue if the number of fragments is still less than a user-defined threshold. This is a simple but efficient check to prevent unnecessary computation. Otherwise, a tree would be added to the queue, extended with another fragment, and immediately discarded since it defies the upper threshold for the maximal number of fragments.

As discussed before in section 6.4.3, terminal fragments have properties of their own, which affect the overall properties of the resulting molecule. For example, if a linker is connected to an oxygen atom, then terminating this linker with a hydrogen atom adds a hydrogen bond donor. This may lead to an exclusion of the molecule if the number of hydrogen bond donors now exceeds the upper bound. However, it would be possible to attach another fragment at the same position that would not result in a hydrogen bond donor. If the fragment tree would be discarded at this point, this valid molecule could not be found. Therefore, the decision to add a fragment tree to the tree queue is based on the properties of a fragment tree *without* terminal fragments. The decision to further process a molecule is based on property values *including* terminal fragments. By considering these two cases separately, potential precursor molecules are not rejected.

When trees are translated into molecules, a copy of the associated fragment is retrieved from the fragment space for each node. The same is done for terminal fragments. Then the tree is traversed and for each edge between two nodes, a bond is created in the respective fragments. For this, the linkers are removed and the bond is connected to the adjacent atoms. After the molecule construction has finished, the linear descriptor for this molecule is generated by NAOMI functionality. The descriptor is then used to query the molecule database to determine whether the molecule has been seen before or not. This step is necessary since the same molecule may have been constructed from a different tree, by exhibiting any of the unresolved reasons for redundancy discussed in section 6.4.5. Due to the differentiation between fragment tree and molecule instance, this does not lead to the exclusion of molecules. If the molecule is unique, its descriptor is added to the molecule database.

Finally, non-monotonously additive properties are calculated and compared to user-defined thresholds. The values are either in range or out of range. Unlike the monotonously additive properties, the non-monotonously additive properties are not considered for the exclusion of fragment trees. The aforementioned tolerance is applied here as well, while all previous validations are also taken into account. If less than k values deviate from the desired range, the molecule fulfills all property requirements. It is subsequently written to the output file as unique SMILES [114].

6.4.8 Alternative Enumeration Strategy

The strategy described above simply uses all fragments as starting points. Although there are use cases where the enumeration of all molecules described by a fragment space is neither necessary, nor desired. In molecular modeling, a certain scaffold or functional unit of a molecule is oftentimes known. This knowledge can be used to pre-select interesting fragments as a way of further restricting the search space. FSees features an alternative enumeration mode, that allows for the selection of such starting fragments by the user. These fragments are then used as starting points only and excluded from being attached during the enumeration. Hence, the resulting molecules are guaranteed to contain exactly one of these fragments. The selection of such starting fragments is used for several experiments discussed in section 8.2.

At this point, there is no explicit mode for building several starting fragments into a single molecule. However, this can be achieved by running the algorithm individually for each starting fragment. In this case, it is necessary to filter redundant molecules and those not made of desired fragments after the enumeration. In order to prevent such a post-filtering, it would be possible to add this and other enumeration strategies in future versions.

7 Data Sets and Parameters

In the context of fragment-based enumeration, several data are required: Molecules and chemical reactions as input for the construction of fragment spaces, physicochemical properties as parameters for the enumeration algorithm, and molecules for validation. This information is either directly or indirectly derived from experiments. Experimental data play a crucial role in the development of novel computational models and algorithms. They are used as a basis for the underlying model or to validate the predicted outcome. Fragment spaces are built utilizing this data in order to create new, meaningful molecules. In this chapter, the different sources of input data are discussed.

7.1 Sources of Molecules

Sets of molecules are required for two purposes. One is the construction of fragment spaces and the other is the validation of enumerated molecules. In the former case, molecules are used as input for fragmentation or reaction-based construction. During fragmentation, molecules are decomposed into their building blocks in order to be assembled into novel molecules. In this case, molecules are used that are drug-sized, bioactive, and non-toxic – if not actual drug molecules – in order to create promising building blocks. For reaction-based construction, small, synthetically accessible molecules are used that must be available via purchase or in-house synthesis. Every molecule is translated into a single fragment. The latter case requires molecules that have a confirmed activity for a desired target. These molecules must be experimentally well-characterized.

A number of resources from which molecules can be downloaded exist. In the following, a description of the datasets and databases that were used as input and for validation is given, including a discussion of which data was retrieved for which purpose. Every set of molecules that was downloaded was subject to a filtering step with NAOMI. This was done by opening the molecules in the NAOMI file format converter [59] and exporting it again. This way, a molecule is read and translated into the NAOMI chemical model. If this molecule contains errors and cannot be read, it is not converted. Due to the canonization procedure in NAOMI [59], molecules that are identical will lead to identical output and can then be filtered out.

7.1.1 ChEMBL

ChEMBL is an open, manually curated database for bioactivity data [126, 127], accessible via https://www.ebi.ac.uk/chembldb/. It contains binding, functional, and ADMET information for many drug-like bioactive compounds. The data are manually extracted from literature, curated, and standardized to make them conveniently accessible. The database contains a range of different drug-types, such as synthetic small molecule, natural product-derived, inorganic, polymer, antibody, peptide/protein, oligonucleotide or oligosaccharide. In this work, version 20 – released on 14th January 2015 – was used. It contains 1.7 million compounds and 13.5 million bioactivity measurements for 10,774 protein targets. The information stored in the database is used in chemical biology, biochemistry, medicinal chemistry and drug-discovery research. ChEMBL is considered an extensive, high quality resource for bioactivity data.

In this project, ChEMBL was queried to retrieve known bioactive molecules for a number of targets. These molecule sets were either used as input for fragmentation in order to create a fragment space or as a reference set to which enumerated molecules were compared.

7.1.2 SureChEMBL

SureChEMBL is an open database containing compounds from patent documents [128]. The data are automatically extracted from literature utilizing text and image-mining techniques. The SureChEMBL database provides a web-based interface for searching molecules based on structure (e.g. substructure or molecular similarity) and for searching the corresponding patent documents. Furthermore, the set of extracted molecules is available for download. Both are accessible via https://www.surechembl.org. In patents, many molecules are described that are not part of any publicly available resource and have only existed in in-house compound libraries. In this work, SureChEMBL was used as a reference to compare enumerated molecules with patented ones. The SureChEMBL dataset included all molecules up to the "Q3 2015" release. This includes 15,592,586 molecules of which 12,790,627 are unique and valid according to NAOMI.

7.1.3 Drugbank

DrugBank is an open database of drug molecules and corresponding protein target information [84], accessible via http://www.drugbank.ca. Drug molecules are divided into "small molecule drug" and "biotech drug" categories. The latter encompasses protein and peptide molecules. Data is manually curated, checked by automatic procedures, and reviewed by experts. The DrugBank interface provides extensive functionality to browse and search drugs according to numerous characteristics, such as name, enzymatic reaction, pathway, chemical class or target; as well as structure, molecular weight, sequence, or associated experimental results, respectively. This is accomplished by associating more than 80 data fields with a single entry. In this project, the "group" assignment is particularly interesting. Group designations are approved, experimental, nutraceutical, illicit, or withdrawn. Drugs in the *approved* group have been approved in at least one country. In this project, version 4.2 of DrugBank was used, which contains 7759 drug entries. These include 6813 entries for small molecules and 1554 molecules in the approved group. After filtering with NAOMI, 1537 unique and valid molecules remained.

7.1.4 DUD-E

The “Database of Useful Decoys: Enhanced” (DUD-E) is a collection of active compounds and decoys for 102 diverse target proteins [129], available from http://dude.docking.org. The active compounds are ligands with experimentally measured activity data that are supported by a literature reference. The authors used data from ChEMBL09 [126] to compile this data set. Since DUD-E was originally developed to benchmark docking algorithms, decoys in this case are molecules that have similar physicochemical properties as the active compounds, but are topologically different so that they most likely do not bind to the target protein. The set contains 50 decoys for each target and a total of 22,886 active compounds. It thus represents a valuable resource not only for docking studies, but for all applications where well characterized bioactive molecules are required. For this project, only the active ligands, also denoted as "known actives", are considered. They are used to construct a fragment space for each of the 102 target classes via fragmentation that are then enumerated.

7.1.5 ZINC

ZINC is an open database containing commercially available compounds [130, 131]. It is intended as a resource for ligand discovery and hence supplies molecules in "popular ready-to-dock formats" [131]. Molecules are presorted into a variety of subsets according to their physical properties, e.g., lead-like, fragment-like, or drug-like; and according to their source, i.e., the vendor that supplies these molecules. The web site – accessible via http://zinc.docking.org – provides an interface to search molecules by a variety of criteria including name, structure, biological activity, physical property, or vendor. Furthermore, it allows users to create their own custom subsets to share or download. The web-based interface was recently updated. For this work, both versions ZINC12 and ZINC15 were used. ZINC contains over 35 million purchasable compounds and represents one of the most extensive resources of directly available molecules.

In this project, the "drug-like" subset containing $1.5 * 10^7$ molecules was used as a reference in the comparison of existing and enumerated molecules; the "lead-like" set comprising $7 * 10^6$ molecules was used to assess the overlap of a large enumerated space with the known chemical space. Furthermore, ZINC was used to download the ChemBridge Building Blocks [102] (see section 7.1.6). All downloaded collections were filtered with the NAOMI converter, so that they only contain unique and valid molecules according to NAOMI.

7.1.6 ChemBridge Building Blocks

The ChemBridge Building Blocks is a collection of molecules available for purchase. It contains relatively small molecules that can be used for the synthesis of larger structures. Use cases include "combinatorial lead generation, hit-to-lead, lead optimization and medicinal chemistry programs" [http://www.chembridge.com/building_blocks/]. In the cheminformatics context, the dataset was used for constructing the SCUBIDOO database [101]. According to the ChemBridge website, the collection contains over 14,000 products. Although there is no direct download option, ChemBridge is a vendor that can be found in ZINC. The ChemBridge Building Blocks collection deposited in ZINC contains 18,053, of which 17,974 remain after uniqueness filtering. In this work, this collection was used to generate a reaction-based fragment space.

7.2 Reaction Sets

Chemical reaction information is required for the construction of fragment spaces. As described in section 4.3, reactions can be used as retrosynthetic rules using a fragmentation approach or directly to annotate synthetic libraries. There are a few sets of retrosynthetic rules that are available for automated processing as well as sets of reactions in machine readable format.

7.2.1 Retrosynthetic Rules

The set of retrosynthetic rules used for RECAP (Retrosynthetic Combinatorial Analysis Procedure) contains 11 chemical bond types corresponding to common chemical reactions [57]. Although RECAP was not directly used during this project, its cut rules are available in Fragment Space Commander (see section 9.1). More importantly, it has inspired the development of BRICS (Breaking of Retrosynthetically Interesting Chemical Substructures) [34].

BRICS attempts to improve RECAP by using a more elaborate set of rules and additional post-processing filters. This is accomplished by taking the adjacent substructure into account for a particular bond type. BRICS defines 16 such substructures in SMARTS pattern language [100]. Each substructure defines one side of the cut bond and is identified with a unique linker name. Since several substructures are compatible, a total of 55 bond types are defined as depicted in Figure 7.1. The complete list of SMARTS patterns is provided in Appendix C. Besides identifying suitable cut positions, these substructures serve as filters to avoid the generation of unwanted chemical motifs and small terminal fragments, i.e., hydrogen and halogen atoms, or hydroxy, nitro, carboxylate, methoxy, methyl, ethyl, and isopropyl groups. Both RECAP and BRICS only define acyclic bonds, so that ring structures are left intact. After fragmentation, fragments with more than 16 heavy-atoms, containing rings with more than eight heavy-atoms, or matching a reactive or toxic group according to Kazius et al. [120] are removed. Furthermore, BRICS defines terminal fragments for each link type (Figure 7.2) as required for the construction of valid molecules discussed in section 3.4. In addition to cut rules, BRICS also refers to three fragment spaces of different size, which are publicly available for download from www.zbh.uni-hamburg.de/BRICS/. However, they were not used in this project. The BRICS fragmentation and filtering procedure was implemented (see section 6.1) and the retrosynthetic rules were used to generate numerous fragment spaces for enumeration (see section 8.2).

7.2.2 Synthetic Reactions

The set published by Hartenfeller et al. contains 58 unique organic synthesis reactions that were specifically compiled for computer-based molecule construction [96]. The reactions are provided as SMIRKS [100] in order to be compatible with a variety of tools and modeling libraries. The set contains 29 *simple* reactions that form one additional bond between reactants. Furthermore, it contains 29

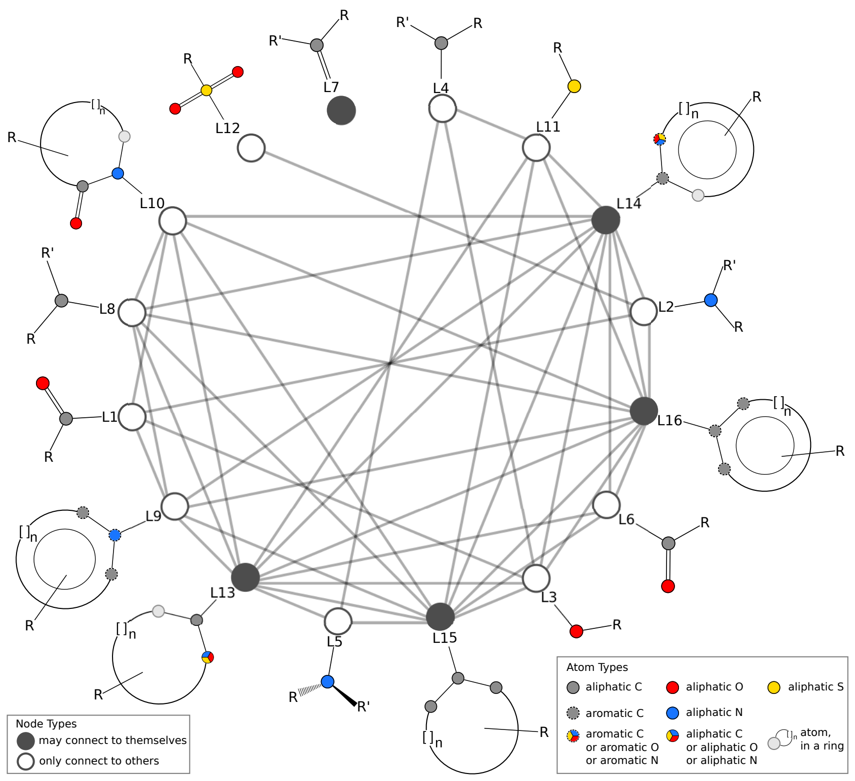


Figure 7.1: Depiction of chemical environments used for the generation of BRICS fragment spaces. L1 through L16 denote (dummy) link atoms used for identification of compatible bond types. Lines between fragments indicate that a bond can be formed between the atoms adjacent to the link atom of each respective fragment, thus removing the link-atom. R-groups are used to describe the diversity of the fragments and may contain additional link atoms. The R groups may also consist of hydrogen only. Fragment depictions were generated with SMARTSViewer [62] *This figure was originally published in [2].*

ring formations that cannot be directly represented by a fragment space. A list of the Reaction SMARTS pattern is provided in Appendix C.3.

7.3 Constraints

The main parameter for the enumeration algorithm are constraints for physico-chemical properties. Meaningful values can be derived from individual reference molecules or molecule collections. These values can be queried from a database or computed with software tools. The values for the individual properties determine which part of the chemical universe is enumerated. Several studies determined

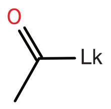
Terminal Fragments	Link types
$H-Lk$	R4 R8 R13 R14 R15 R16
H_3C-Lk	R1 R3 R5 R6 R9 R10 R11 R12
$H_2C=Lk$	R7
	R2

Figure 7.2: List of BRICS terminal fragments [34] and the respective link types.

interesting chemical spaces through statistical analysis of large molecule sets [12–14, 16–19, 53] as discussed in section 3.2. The thresholds for two of these spaces are used as enumeration constraints, thus creating only molecules belonging to these spaces. The first is the drug-like chemical space as defined by Lipinski et al. [15], also known as *Rule of five* (see section 3.2). The second was derived from lead-like chemical space. In the experiments, lead-like constraints as defined by ZINC (*lead-like$_Z$*) were used, which are directly derived from Oprea's lead-like definition (*lead-like$_O$*) described in section 3.2 [14]. Table 7.1 lists the physicochemical properties used and their values.

Table 7.1: Physicochemical constraints for lead-like and drug-like molecule classes. Lead-like$_O$ describes constraints according to Oprea et al. [14] and lead-like$_Z$ according to ZINC [131].

	drug-like	lead-like$_O$	lead-like$_Z$
Molecular Weight	≤ 500	≤ 450	250 – 350
clog$P$	≤ 5.0	-3.5 – 4.5	≤ 3.5
H-bond Donors	≤ 5	≤ 5	
H-bond Acceptors	≤ 10	≤ 8	
Rotatable bonds			≤ 7
Non-terminal single bonds		≤ 10	
Rings		≤ 4	

8 Evaluation and Experiments

This chapter describes the experiments carried out with the two newly developed algorithms for reaction-based construction and enumeration of fragment spaces and discusses the results. Since FSees was developed first and experiments were done based on fragment spaces from retrosynthetic fragmentation, it is discussed first. The first section describes the optimization of the enumeration strategy. Then, various enumeration experiments are discussed. Finally, experiments for the construction of reaction-based fragment spaces are presented.

8.1 Optimization of Enumeration

The enumeration algorithm is the result of a continuous optimization process that has accompanied the development of the algorithm. In this section, strategies relevant to runtime are discussed. All computations were carried out on a workstation with an Intel i5 processor (i5-4570 CPU @ 3.20GHz) and 16 GB RAM. The database files were stored on the internal hard drive rather than the network file system and no other disk or CPU intensive jobs were running during computations.

8.1.1 Prioritization of Filters

This investigation was carried out to show the effect of the computation of $\log P$ on the performance of the enumeration. Three different sets of constraints were used, based on drug-like values: with log$P$, without log$P$, and with a *rotatable bonds* cutoff instead of log$P$. The first constraint triggers the relatively costly computation of log$P$, the second only uses monotonously additive properties (see Definition 6.1), and the third substitutes log$P$ with a less costly computation. The same study was carried out for three different fragment spaces for DUD-E targets [129] Androgen Receptor (ANDR), Hexokinase type IV (HXK4), and Thymidylate synthase (TYSY) (see section 7.1 and 8.2.3). The results are shown in Figure 8.1. Since the different parameter sets result in different numbers of molecules, the absolute runtimes cannot be compared so that the average rate with which molecules are generated is used instead.

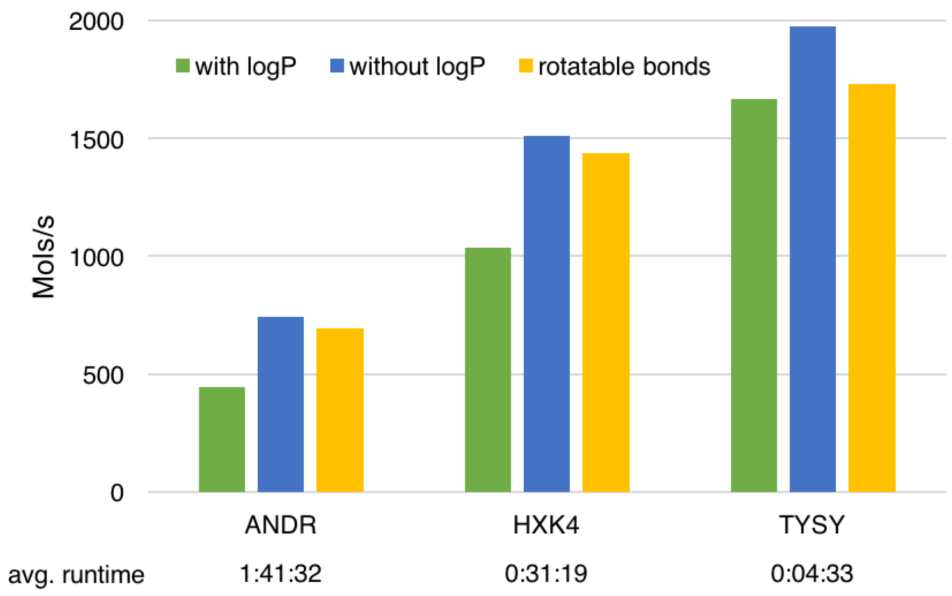


Figure 8.1: Average rate of molecule generation for three different fragment spaces (ANDR, HXK4, and TYSY) with three different sets of parameters (with log$P$, without log$P$, and rotatable bonds). The average runtime for each fragment space is given on the bottom in h:m:s.

The results are very similar for all three fragment spaces. The constraint set *with logP* (green) results in the slowest rate, while the set *without logP* (blue) results in the highest rate. The rate for the set including *rotatable bonds* (yellow) is in between the first two. In two cases, ANDR and HXK4, it is almost as high as the rate for *without logP*. The difference in the rates between fragment spaces arises from the different runtimes. At the beginning, the rate is usually higher since a lot of new molecules are generated (see also 8.1.3) and shorter enumerations consequently have higher average rates. The difference between the highest and lowest rates is much more distinct for longer running enumerations. For

ANDR, the lowest and highest rates are 446 Mols/s and 740 Mols/s, respectively. This represents a 65% increase. For TYSY, the increase is only 18% (from 1666 and 1975 Mols/s).

Two conclusions can be drawn: 1. The computation of a non monotonously additive property affects the performance. In the case of log$P$ this effect is much more significant than in the case of rotatable bonds. 2. The longer the runtime, the stronger the effect because proportionately more molecules must be processed.

8.1.2 Performance Improvements

When new strategies to increase performance were introduced, their effect on the runtime was measured. Improvements include the adjustment of the algorithmic procedures and database parameters. Results for two fragment spaces from the DUD-E data set are shown, i.e., insulin-like growth factor I receptor (IGF1R) and TGF-beta receptor type I (TGFR1). The effect of algorithmic strategies are depicted in Figure 8.2, of database parameters in Figure 8.3. The former were implemented iteratively, so that each step includes the changes from the previous.

Baseline represents the starting configuration, i.e., the initial implementation with file-based data structures and redundancy prevention according to Pärn et al. (see section 6.4.5). The first optimization was implemented by sorting fragments according to their molecular mass (*Sorted fragments*). This stops the extension of a fragment tree as soon as the first fragment is attached that results in a higher molecular weight than the respective upper bound. As a second improvement, a buffer was added to the tree and molecule databases. This way, several elements are added to the database at once, rather than having a separate transaction for each newly added molecule or tree. Different buffer sizes were tested with the capacity to store 10^3, 10^4, and 10^5 elements, respectively. Although a buffer size of 10^5 performed best for TGFR1, this could not be observed in other cases (e.g. IGF1R); therefore, a buffer size of 10^4 is used. The *Symmetry Rule* is described in detail in section 6.4.5. It prevents symmetrical fragments to cause redundant molecules. The effect of this change is quite different for the two fragment spaces shown since it strongly depends on the presence of symmetric fragments. The different performance improvements in the cases of IGF1R and TGFR1 show that the latter contains fewer symmetric fragments than the former.

The last test was carried out to show the effect of the Bloom-filter, which was added at an early stage of the project and has therefore been in use during the previous tests. Not using the Bloom filter results in a significant increase in runtime, i.e, 45% (108 min.) in the case of IGFR1 and 22% (110 min.) in the case of TGFR1.

In addition to algorithmic improvements, the database parameters for the fragment tree and molecule storage were optimized. The SQLite implementa-

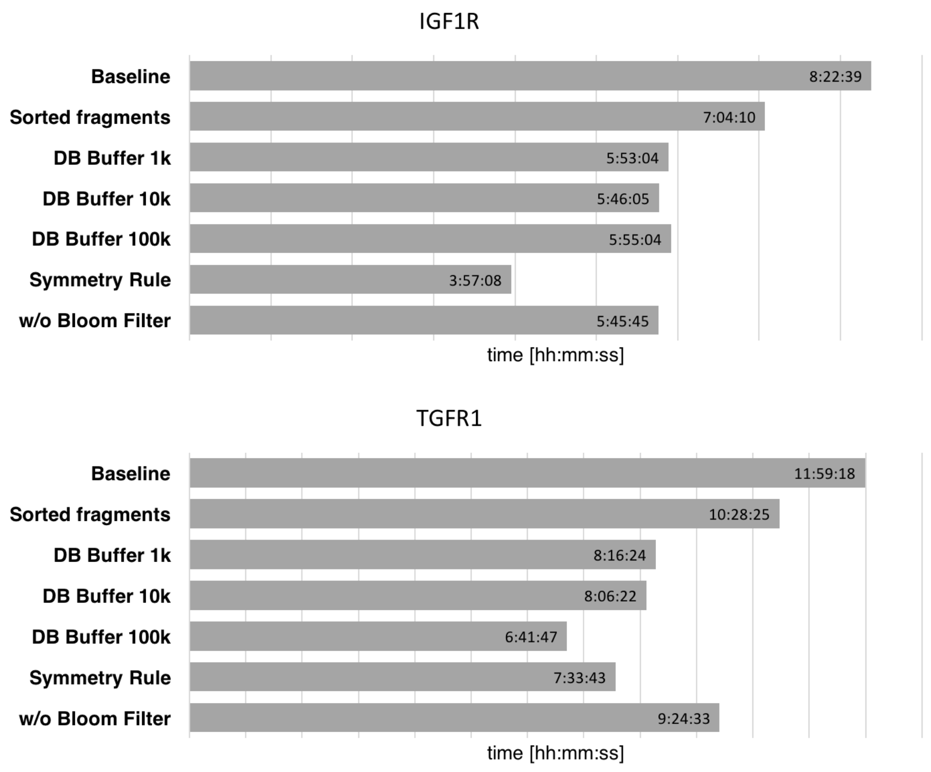


Figure 8.2: Runtime of enumeration runs for DUD-E targets IGF1R and TGFR1 with different optimization strategies. Please refer to section 8.1.2 for details.

tion allows for a user to configure a vast number of parameters [123], some of which significantly affect database performance. For the initial implementation, database parameters were set to reasonable values. The systematic investigation of the effect of individual parameters was done after all algorithmic optimization strategies had been implemented. The systematic optimization of parameters resulted in an additional decrease of runtime by 34% (79 min.) for IGF1R and 36% (165 min.) for TGFR1. These values were derived from a comparison of the perfromance after algorithmic optimization, i.e., *Symmetry rule* from Figure 8.2, and after database optimization, i.e., *best combined* from Figure 8.3.

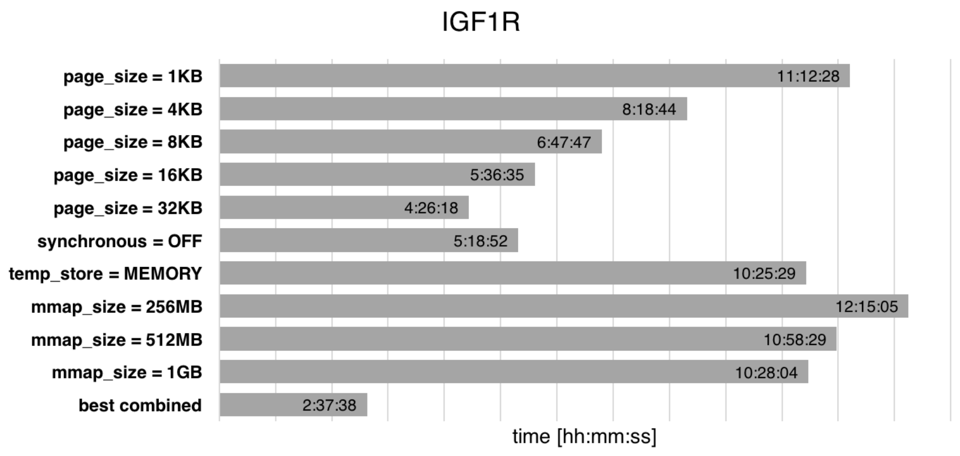
In each test, the default database parameters were used, except for the parameter tested. Default values are:

- page_size = 1024
- synchronous = FULL
- temp_store = FILE
- mmap_size = 0

The first value, a page size of 1 KB (1024 byte), represents the starting point of the optimization (see Figure 8.3). For the last test the best values of each parameter were combined. This represents the final configuration used in the enumerator.

A page is the smallest unit of data in a database, the *page_size* therefore determines its size in bytes. Adjusting this parameter had one of the largest effects on runtime as shown in Figure 8.3. A similar decrease in runtime resulted from the adjustment of the *synchronous* flag. It determines how data is written to the hard drive. In the default mode (FULL) it is ensured "that all content is safely written to the disk surface prior to continuing. This ensures that an operating system crash or power failure will not corrupt the database" [123]. With *synchronous = OFF*, "SQLite continues without syncing as soon as it has handed data off to the operating system." [123]. In this case, the database may become corrupted if the computer crashes. This is not an issue since the database is only created for a single enumeration. If the enumeration must be restarted, a new database instance is created. The *temp_store* parameter determines where temporary tables and indices are kept. The value *MEMORY* specifies that they are stored in memory rather than on the hard drive. For TGFR1 this parameter leads to a significant increase in runtime compared to the initial test. In this case, the process runs out of main memory so that the linux operating system starts "swapping". Finally, *mmap_size* determines "the maximum number of bytes of the database file that will be accessed using memory-mapped I/O" [123]. Although the last parameters increases the runtime in the case of TGFR1, it proved to decrease the runtime for shorter running simulations, such as IGF1R (6% decrease). However, due to its small impact at a high value it was not used.

One more parameter that is always used is the size of the database cache, *cache_size*. This has been implemented as a user-definable parameter that can be adjusted according to the available main memory. In the previous tests it was set to 1024 MB.



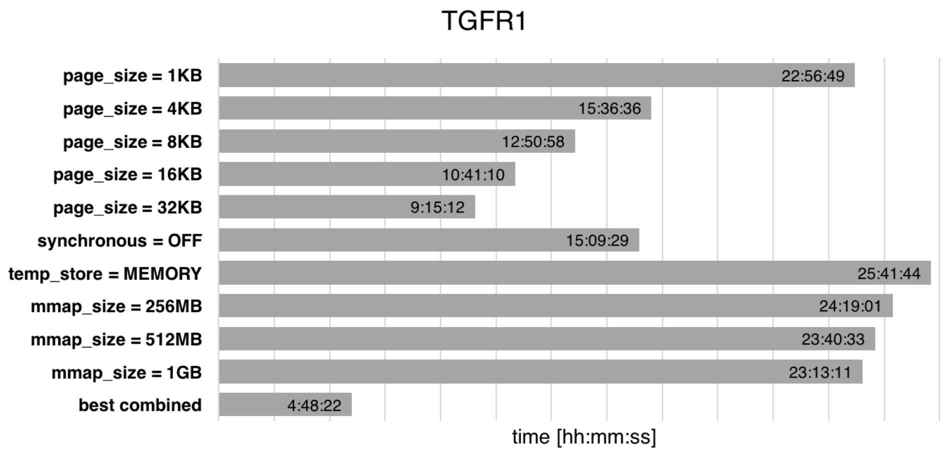


Figure 8.3: Runtime of enumeration runs for DUD-E targets IGF1R and TGFR1 with different database parameters. Please refer to section 8.1.2 for details.

8.1.3 Memory Consumption

The FSees algorithm is designed to be resource-efficient, especially with respect to main memory. This is mainly accomplished by using file-based data structures based on SQLite. Some memory is still required, but it can be limited by setting the appropriate parameters. There are two customizable parameters that affect the amount of memory used, the database cache and the Bloom-filter size. Both values can be set at the beginning of the enumeration and must be specified in MegaBytes (MB).

In the following test, the database cache was set to 1024 MB and the Bloom filter to 1280 MB (10^{10} bits) per database. This results in a memory requirement of 5.5 GB for the databases. In addition, the process itself requires memory for certain permanent and temporary instances, e.g., the fragment space, fragments, product trees, molecules, descriptor strings, database buffers, etc. The behavior of the algorithm regarding memory consumption and storage utilization is investigated. Therefore, the fragment space for the DUD-E target Beta-2 adrenergic receptor (ADRB2) was enumerated and the result is shown in Figure 8.4. Two more examples, FA7 and FABP4, are shown in Figure 8.5.

All curves show a distinct progression that is similar in all examples. At the beginning of the process, the main memory consumption (grey) rises steeply until it levels off around 5.5 GB, as expected. The memory usage of non-database instances is negligible. As the algorithm progresses, each fragment is used as a starting point. After a starting fragment has been completely explored, the tree database is emptied since new trees cannot contain fragments previously used as starting points. Therefore, trees from previous iterations are discarded to save memory and speed up the tree database. This is reflected in the progression of the tree database fill state (green) while the tree queue shows the complementary behavior (brown). The tree database grows until all possibilities for extension have been explored. It is then emptied and the curve drops to zero. The tree queue reaches its maximal fill state quickly and decreases when less new trees are generated. The last two curves show the number of molecules (blue) generated and the corresponding rate in molecules per second (purple). The number of molecules increases steadily, while the slope of the curve decreases slightly. The curve of the rate behaves complementarily; it is very high at first but then decreases slowly. The longer the enumeration runs, the more molecules have been generated and stored in the database. Consequently, queries to the molecule database, i.e., redundancy checks, take more time.

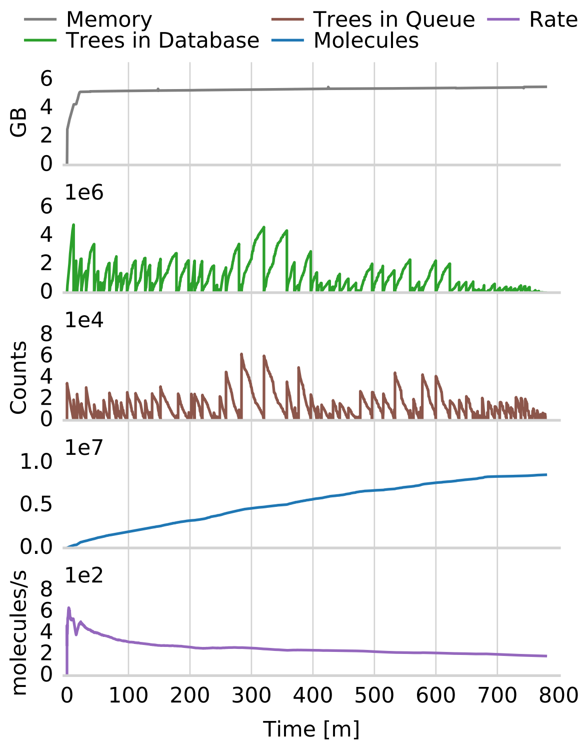
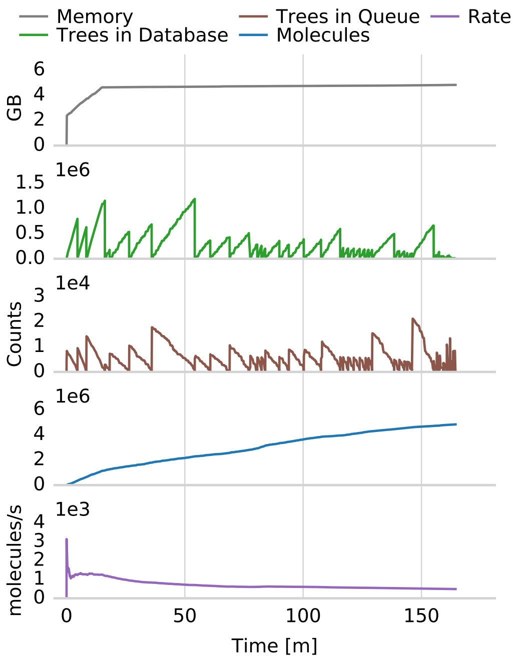
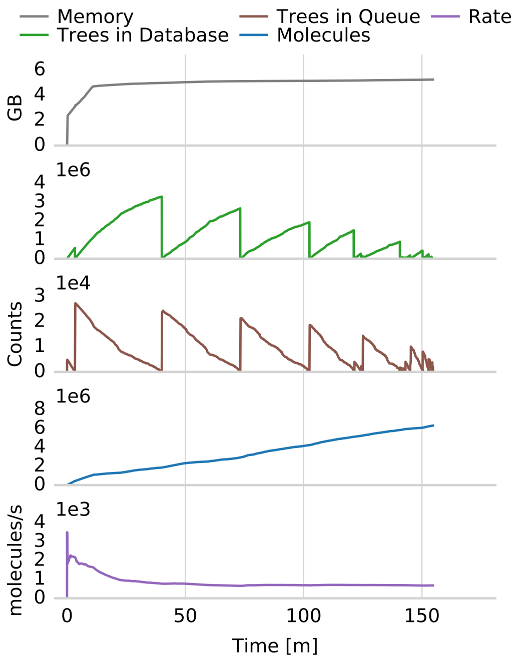


Figure 8.4: Progression of different resources during the enumeration process of target ADRB2 of DUD-E dataset. From top to bottom: Memory consumption, tree database, tree queue, molecule database, and rate of generated molecules. *This figure was originally published in [1].*



(a) FA7



(b) FABP4

Figure 8.5: Progression of resources during the enumeration of fragment spaces for (a) Coagulation factor VII (FA7) and (b) Fatty acid binding protein adipocyte (FABP4) of the DUD-E dataset. See also Figure 8.4.

8.2 Enumeration Experiments

This section describes several enumeration experiments. For this purpose, fragment spaces of various sizes were created and different sets of enumeration constraints were used. In a large scale experiment, 102 fragment spaces were created and then enumerated with individual constraints. They are each based on one target class of the "Database of Useful Decoys: Enhanced" (DUD-E) [129]. This dataset was described in section 7.1. Then, potential kinase inhibitors and microtubule-destabilizing anticancer agents were generated. In these cases, an alternative enumeration strategy is applied focusing on particular *fragments of interest* demonstrating how the enumeration algorithm can be applied to typical use cases of fragment-based design. Finally, another enumeration strategy is applied to enumerate a very large fragment space derived from approved drug molecules. In this case 0.5 billion lead-like molecules, named Hamburg enumerated lead-like set (HELLS), were generated.

8.2.1 Experimental Setup

The overall experimental setup is similar for all enumerated fragment spaces. First, a fragment space was created by retrosynthetic fragmentation of known active molecules for each target class. Then, each fragment space was enumerated. Therefore, either target-specific constraints were used or general-purpose constraints, such as drug-like [15] or lead-like [12, 14] (see section 7.3). Finally, the resulting molecules are subject to analysis.

The lead-like chemical space contains molecules that are usually used in earlier phases of the drug-discovery pipeline because they are relatively small. This leaves much room for the optimization of a structure. Thus, these constraints are used for the generation of a large set of molecules that can be used with other lead-finding technologies. Drug-like constraints may be more interesting at later stages, e.g., when optimizing a structure. It can be used to find molecules with the same physicochemical properties but different scaffolds or decorations. Since the drug-likeness filter employed here only defines upper bounds that are relatively large, it may lead to a great number of molecules in a wide range of sizes. In practice, it is better to use constraints specific to the target in order to allow for a tailored library, a manageable amount of enumerated molecules, and a reasonable runtime. Hence, the majority of constraints are derived from properties of known active molecules, oftentimes the same molecules used for the construction of the fragment space. Due to the fact that these molecules already exhibit an activity for a certain target, the assumption is that newly generated molecules show similar behavior with respect to activity and bioavailability. Drug-like constraints as defined by the *Rule of five* [15] were only used for one experiment. Most other constraint sets are target specific while using the same descriptors as drug-like, i.e., molecular weight, number of H-bond donors and acceptors, and cLogP.

Besides physicochemical constraints, the maximal number of fragments was limited in all experiments to further limit the search space. This parameter greatly affects the number of resulting molecules due to combinatorics. Furthermore, the database cache was set to 3 GB, i.e., 1 GB per database instance, in all experiments. For the DUD-E and HELLS enumeration, the size of the Bloom filter was set to 5 GB ($4 * 10^{10}$ bits) and to 1.25 GB (10^{10} bits) for target-specific experiments.

8.2.2 Evaluation Strategy

In order to evaluate a large library of enumerated molecules and asses their quality, an easy to apply, fast to compute, objective, and reproducible approach is required (see introduction of chapter 5). Here, each generated library was compared to known bioactive molecules. Individual molecules were compared based on Tanimoto similarity of the Extended Connectivity Fingerprint (ECFP) descriptor [111] (see section 5.2). A diameter of four bonds was used in all calculations (ECFP_4). The ECFP is well suited for large scale comparison since it is quick to calculate. Pairwise similarities were calculated and the resulting similarity distributions were analyzed according to the experiment.

8.2.3 DUDE Enumeration

For this large scale experiment, the "known active" molecules for each target of the DUD-E dataset (see section 7.1) were fragmented in order to generate 102 target specific fragment spaces. Here, the BRICS set of retrosynthetic rules [34] was used (see section 7.2). For each target class, the physicochemical property constraints were derived from the input molecules using the same descriptors as drug-like. Therefore, the values of the lower and upper quartile of the property distributions for molecular weight, number of H-bond donors and acceptors, and cLog$P$ were used. The individual values for all classes are listed in Table B.2 in Appendix B. The maximal number of fragments was set to four. This was estimated based on the average molecular weight of all fragments, which is 120 Da. Connecting four average fragments results in a molecular weight of almost the upper bound for drug-like, i.e., 500 Da. In addition, a runtime constraint was applied. The computing time on a single core was limited to seven days to avoid the generation of vast amounts of molecules. Yet, the algorithm is able to enumerate large chemical space and run for a long time; oftentimes it is neither reasonable nor necessary to generate a great number of molecules because working with them can be tedious. For these long running cases it is recommended to either revise the underlying fragment space by reducing the number of fragments, or to further limit the search space by applying more stringent constraints.

As a result, eight of the 102 computations were terminated after 7 days, i.e., EGFR, FA10, HIVPR, MK14, PGH2, SRC, TRY1, CP3A4 (for the full names please refer to Table B.1). Until this point, between $4.1{*}10^7$ and $1.1{*}10^8$ molecules had been generated. Figure 8.6 shows a summary of the 94 successfully finished enumerations, a list of the individual results is shown in Table B.4 in Appendix B.

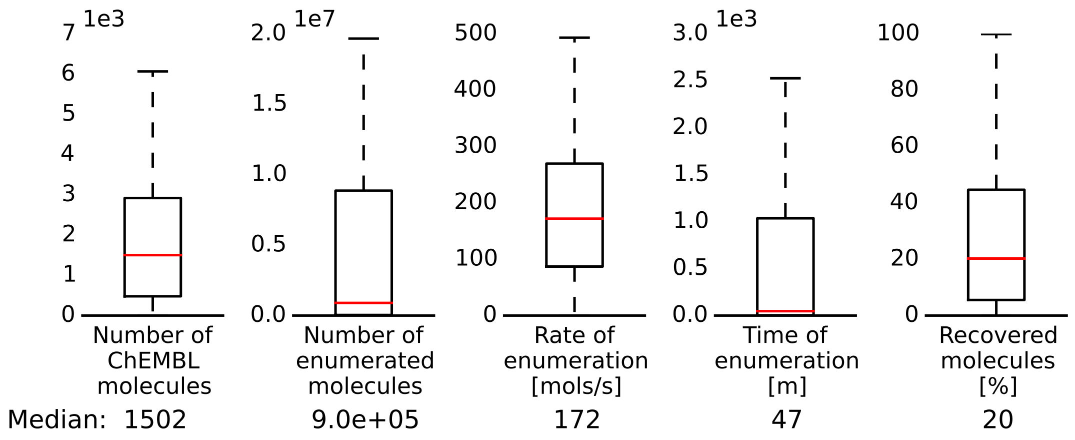


Figure 8.6: Distributions of results from DUD-E experiment as boxplots (outliers omitted). From left to right: Number of bioactive molecules in ChEMBL sets, number of enumerated molecules, average rates of individual enumerations, computing times of individual enumerations, percent of recovered molecules used for the construction of a fragment space. *This figure was originally published in [1].*

Two targets, PUR2 and SAHH, were excluded at this point since these enumerations resulted in 0 and 2 molecules, respectively. The fragment spaces only contained 16 and 42 fragments, respectively. Although this is a relatively small number of fragments it is not too small. The reason for not retrieving more molecules was the set of constraints (see Table B.2). In the case of PUR2, molecules with exactly 11 H-bond acceptors and in the case of SAHH, molecules with exactly 4 H-bond donors, were requested. These constraints are too stringent to enumerate with such a small number of fragments. The reason for these values are the composition of initial molecules. In both cases they are all very similar and show a very narrow distribution for the properties in question.

The enumerated molecules were compared to known inhibitors from ChEMBL [127]. Therefore, molecules were retrieved from ChEMBL first. Each target PDB ID was manually mapped onto a ChEMBL target ID for this purpose (see Table B.3). Five targets could not be found in ChEMBL, i.e., ADRB1, HIVINT, HIVPR, HIVRT, and INHA. All corresponding bioactive molecules were downloaded via the ChEMBL Web Services, which provides access to the database via web request. It returns XML files that were processed with Python.

For the remaining 88 targets, the "known actives" were removed from the ChEMBL set. This was done in order for the ChEMBL set to not contain molecules that were used in the construction of the fragment space. The goal is to investigate how similar enumerated molecules are to known bioactive molecules and whether *new* promising molecules can be found. Initial molecules that were generated during enumeration would cause similarity of 1.0 and therefore create an unwanted bias.

After assembling the necessary data for each target, the ChEMBL compounds for each class were used as a reference set. All enumerated molecules were compared to all ChEMBL compounds of the respective class. The same computation was carried out for the "drug-like" subset of the ZINC database [131] The ZINC set contains $1.5 * 10^7$ molecules[12] and was compared with the target specific sets of ChEMBL compounds. The goal is to assess the performance of the enumerated library compared to a library of available molecules. Both comparisons – enumerated vs. ChEMBL and ZINC vs. ChEMBL – yield similarity distributions that are compared as follows: A receiver operating characteristic (ROC) curve was computed to assess whether an enumerated library or the ZINC set is more similar to the respective ChEMBL subset. Each bin of the similarity distribution contains the number of molecules in a certain similarity range, e.g. [0.1..0.2). To compute the ROC, for each similarity bin, the number of molecules in the range from the current bin i to N (where N is the number of bins) was computed. The values were stored as EC_i for enumerated vs. ChEMBL and ZC_i for ZINC vs. ChEMBL. The values of EC_i and ZC_i were plotted onto the X and Y axes, respectively (see Figure 8.7 and B.3) Then, the area under the curve (AUC) was calculated by means of the following formula:

$$AUC = \sum_{i=0}^{N-1} EC_i * |ZC_{i+1} - ZC_i|$$

In the following analysis, the resulting AUCs were classified into three ranges: $[0.0 - 0.4)$ for "ZINC more similar to ChEMBL", $[0.4 - 0.6]$ for "no clear distinction", and $(0.6 - 1.0]$ for "enumerated more similar to ChEMBL".

Figure 8.7 shows the similarity distribution and the ROC curve for two targets, i.e., MAP kinase-activated protein kinase 2 (MAPK2) and Cytochrome P450 2C9 (CP2C9). The AUC for MAPK2 is 0.91 and 0.39 for CP2C9, hence the enumerated molecules of MAPK2 are more closely related to ChEMBL-MAPK2 than the molecules from the ZINC set. For CP2C9, the situation is reversed because the enumerated set is less similar than ZINC. This analysis was applied

[12] Accessed on September 15th, 2015

to all 88 target classes, see Figure B.2 and B.3. There are 78 cases where the enumerated sets had an AUC above 0.6, 2 cases had an AUC below 0.4, and 8 cases where the AUC was in the 0.5 ± 0.1 range.

The two cases that showed weak performance are Thymidylate synthase (TYSY) and Cytochrome P450 2C9 (CP2C9). There are two reasons for this in the case of TYSY: One is the choice of enumeration parameters, most importantly the molecular weight range of 427 to 533. Only 11 of the 64 ChEMBL-TYSY actives are in this range, while most ChEMBL molecules are smaller and – as a result – have less features in the ECFP. The reason for this is that most of the DUD-E actives cover a different molecular weight range than ChEMBL-TYSY, as shown in Figure 8.8. Since the constraints are derived from DUD-E, the enumerated molecules exhibit different properties than the ChEMBL actives. Besides, only two molecules were removed from the ChEMBL set that were also present in the DUD-E set. The second reason for the weak performance is the composition of the fragment space. Most ChEMBL-TYSY molecules exhibit a distinct pattern, namely a ring system with two aromatic rings and either a six-ring with one Oxygen atom or some similar ring (see Figure 8.9). This pattern is not present in the fragment space, neither as fragment nor as part of a larger fragment, and can therefore not occur in molecules generated from this space. Since this pattern leads to distinct values in the ECFP, its absence also severely affects the molecular similarity.

The reason for the bad performance of CP2C9 is not as clear. First of all, the ChEMBL-CP2C9 set contains more than 21,000 molecules and therefore covers a much larger chemical space than most of the other classes (see "ChEMBL" in Figure 8.6). This is due to the nature of Cytochrome P450 enzymes since they bind a variety of molecules as substrate, activator, and inhibitor. The most plausible explanation is that the large amount of diverse molecules prohibits a clear signal.

In the previous analysis, the ROC curve was used to compare two distributions. For this, the whole range of similarities was taken into account. However, one should not overlook that in the similarity range of 0.5 to 1.0, the number of enumerated molecules is often distinctly higher than the number of ZINC molecules (see Figure 8.7). In the case of MAPK2, the ratio is 0.055 for enumerated molecules and 0.002 for ZINC molecules. In addition to the desired physicochemical properties, these molecules also have a significant topological similarity to known and active molecules as defined by the ECFP. This implies that the enumerated set contains numerous promising molecules, which makes it an interesting input for techniques that require molecular libraries, such as virtual screening.

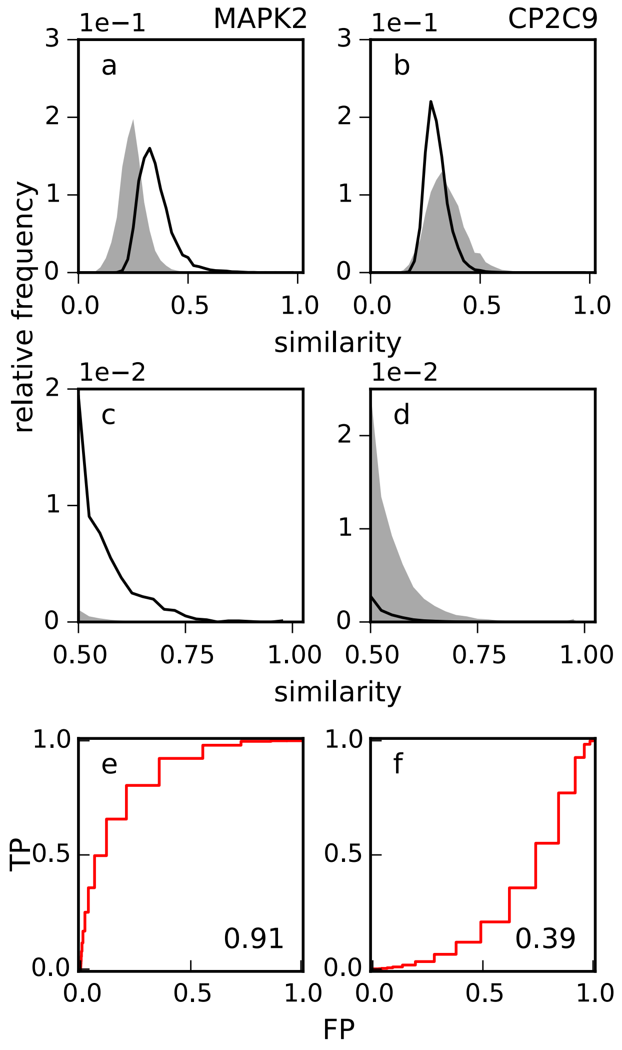


Figure 8.7: Two examples for results from the analysis of the DUD-E experiment showing a favorable (MAPK2, left column) and unfavorable (CP2C9, right column) case. Plots a-d show the distribution of similarity values for ranges 0.0 to 1.0 (a, b) and 0.5 to 1.0 (c, d). The black line shows the similarity distribution of the enumerated molecules versus the ChEMBL set, the gray area the similarity distribution of ZINC versus ChEMBL. Plots e and f show the corresponding ROC curve for the similarity distribution and the AUC as numeric value. *TP* denotes the case where more enumerated molecules have a higher similarity to ChEMBL, *FP* the case where more molecules from ZINC have a higher similarity to ChEMBL. *This figure was originally published in [1].*

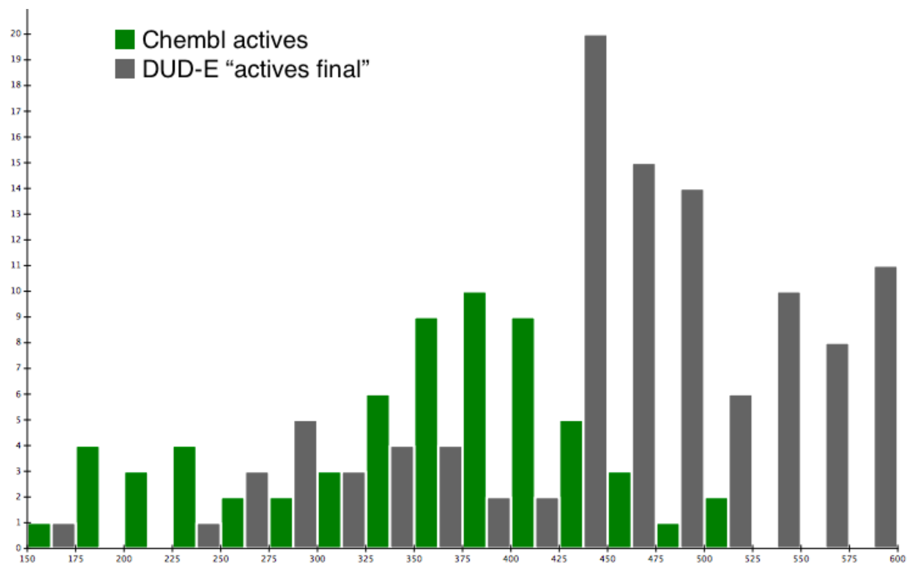


Figure 8.8: Histogram of the Molecular Weight distribution of the "actives final" and ChEMBL bioactives for thymidylate synthase (TYSY) from DUD-E.

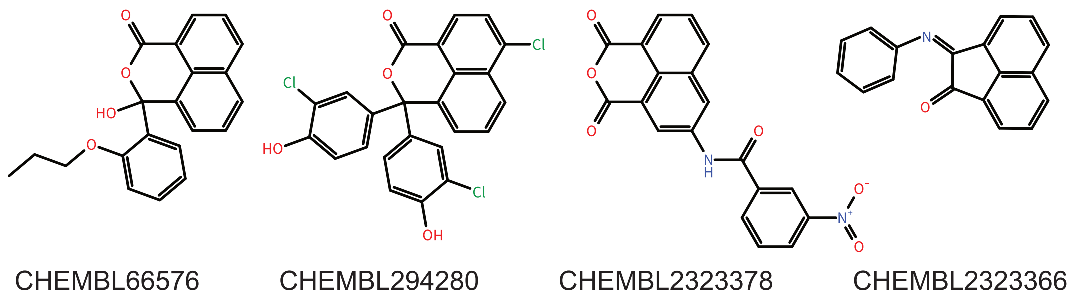


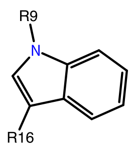
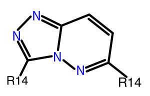
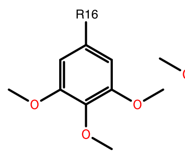
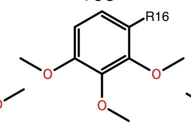
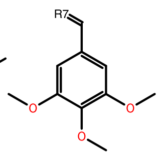
Figure 8.9: Four bioactive molecules from the ChEMBL-TYSY set (Thymidylate synthase).

8.2.4 Specific Targets

The two experiments discussed next describe different scenarios encountered in fragment-based drug discovery. For this purpose an alternative enumeration protocol is utilized (see section 6.4.8). The experimental setup is very similar to the DUD-E large scale experiment except that enumeration focuses on specific *fragments of interest* here. The first experiment is the enumeration of a core fragment with multiple linkers and the second the extension of a fragment with a single linker. The use cases are the enumeration of a class of molecules that exhibits a central fragment as a functional feature and the exploration of possibilities after an initial hit fragment was identified by fragment screening, respectively.

For this experiment, fragment spaces were constructed from ChEMBL actives. The list of known bioactive molecules for the corresponding class were downloaded, filtered for duplicates, and fragmented according to BRICS fragmentation rules [34].

Table 8.1: Depiction of fragments of interest and results of the enumeration experiments for the serine/threonine-protein kinase and Tubulin. *This table was originally published in [1].*

	Serine/threonine-protein kinase		α- and β-tubulin		
Number of fragments	242		165		
Fragments of Interest					
	F196	F199	F58	F61	F64
Enumerated Molecules	2,701,282	2,859,468		575,864	
Computing time (hh:mm:ss)	02:47:18	01:24:09		00:07:21	

Serine/Threonine-Protein Kinase Inhibitors

The first target is a kinase, more precisely a *serine/threonine-protein kinase*. Kinases are enzymes that add a phosphate group (PO_4^{3-}) to a protein or another organic molecule. In the case of proteins, this *phosphorylation* leads to an activation in most cases, sometimes to inactivation. Kinases play a critical role in cellular signaling and thus represent an interesting target for the treatment of cancer. There are two main classes of protein kinases, one adds phosphate groups to a serine or threonine side chain (serine/threonine-protein kinase) the other to a tyrosine side chain (tyrosine kinases). Kinases constitute important targets and are subject to intense pharmacological research. Protein kinase inhibitors are usually relatively linear molecules that often contain a distinct bi-

Table 8.2: Enumeration constraints for serine/threonine-protein kinase experiment.

Molecular Weight	logP	Donors	Acceptors
250 – 500	1.0 – 5.0	1 – 4	5 – 9

or trycyclic fragment [132–135]. This experiment demonstrates how the FSees enumeration algorithm is used to exhaustively enumerate all molecules containing such a fragment. The protein targets considered for this experiment are the closely related serine/threonine-protein kinase 32A, 32B, and 32C from *Homo sapiens* (CHEMBL6150, CHEMBL5912, and CHEMBL5405, respectively). All bioactive compounds were downloaded from ChEMBL, merged, and duplicates were removed. A fragment space with 242 fragments was created from 131 unique molecules (see Figure 8.10). Nine of these molecules were not processed because no BRICS rule could be applied.

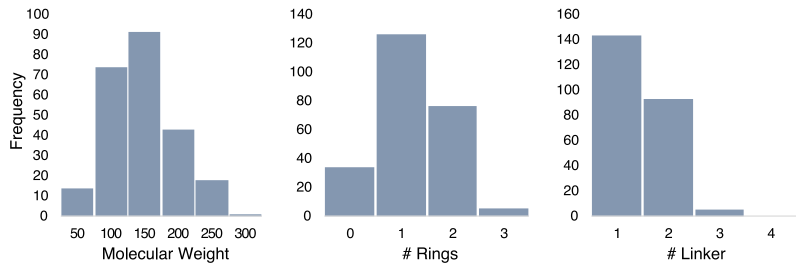


Figure 8.10: Property distributions of the serine/threonine-protein kinase fragment space.

Two fragments were selected for enumeration, F196 and F199 (see Table 8.1). They represent good scaffold fragments since they are relatively large and inflexible due to the ring system and have two linkers where they can be extended. As before, enumeration constraints were derived from the properties of the initial molecules (see section 8.2). The enumerations yield $2.7 * 10^6$ and $2.8 * 10^6$ molecules, respectively. Table 8.1 shows the precise number and the computing times. The property distributions of the enumerated molecules are shown in Figure 8.11 All molecules are in the desired range.

The resulting molecules were then compared to the initial ChEMBL actives by ECFP-Tanimoto similarity. In this case, no background distribution as for the DUD-E experiment was generated. The resulting similarity distribution for fragments F196 and F199 is depicted in Figure 8.12. There are 2296 molecules with similarity values greater than 0.5 for fragment F196 and 2463 for fragment F199. Only one molecule has a similarity value of 1.0 meaning that it is identical to an input molecule (CHEMBL2425646). It contains fragment F196 al-

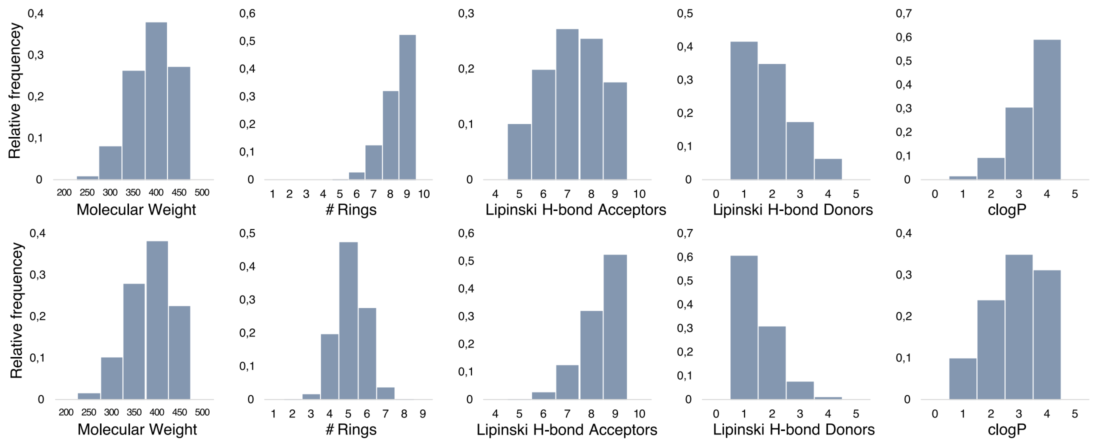


Figure 8.11: Property distributions of enumerated Kinase molecules. Results for fragment F_196 on top and F_199 on bottom.

though, interestingly, it is not the molecule from which fragment F196 was derived (CHEMBL300138). In the enumerated molecule, fragment F196 is connected only via the linker adjacent to the aromatic nitrogen (R9); the linker adjacent to the aromatic carbon (R16) was terminated with a hydrogen (see Table 8.1). At this point, it should be mentioned that fragment F196 with this linker configuration is only present in one initial molecule, while the *indole* pattern is found in 14 molecules.

None of the other molecules in the ≥ 0.5 range exceeds a similarity of 0.8, which is partially due to the fact that the core fragments with this linker configuration are underrepresented in the initial set. Both fragments occur in only one molecule each. However, the other molecules still have significant structural similarity to initial molecules and may represent interesting new alternatives to these inhibitors. The five highest scoring molecules for fragments F196 and 199 with the corresponding bioactive molecules from ChEMBL are shown in Figure 8.13 a, b. The fact that there are no high similarity values observed does not represent a bad result. It merely means that there are not many highly similar molecules in the set to which the enumerated molecules were compared. Whether the result is satisfying depends on the application and the goal of the enumeration. If, for instance, the goal is to enumerate a chemical space of underrepresented inhibitors, this must be seen as a success.

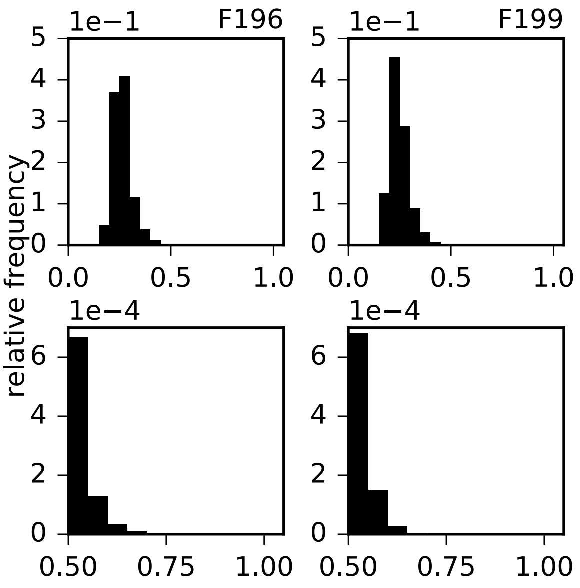


Figure 8.12: Similarity distribution of enumerated molecules compared to bioactive serine/threonine-protein kinase binders from ChEMBL for different starting fragments. Results for fragment F196 are shown on the left, for fragment F199 on the right. First row shows similarity values from 0.0 to 1.0, second row shows range 0.5 to 1.0. *This figure was originally published in [1].*

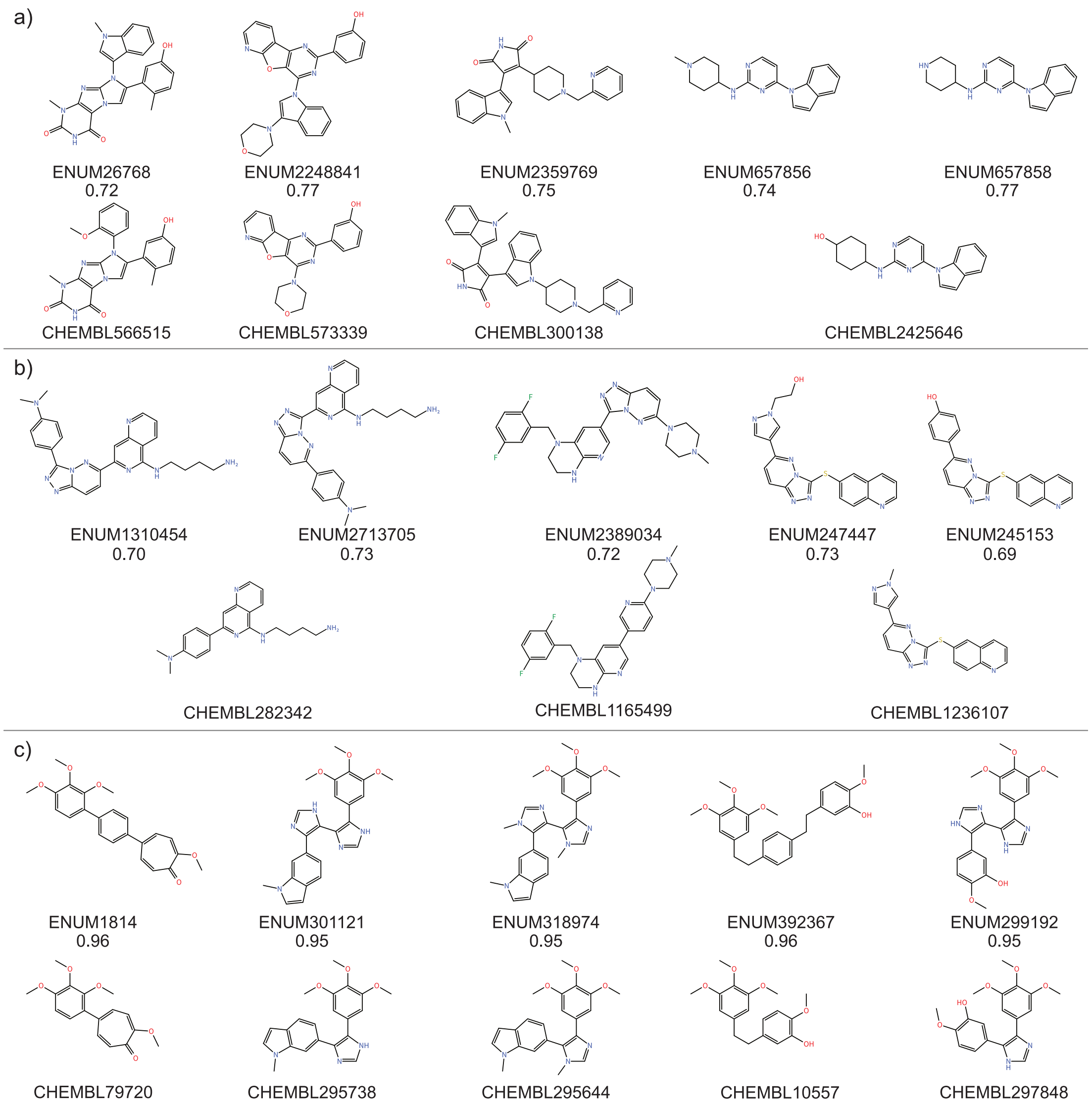


Figure 8.13: The five highest scoring enumerated molecules that are not identical to input molecules for a) Serine/threonine-protein kinase fragment F196, b) fragment F199, and c) Tubulin. The top row of each subfigure shows enumerated molecules, bottom row shows the corresponding bioactive molecule from ChEMBL. The number indicates the Tanimoto similarity of ECFP_4 between the enumerated molecule and the CHEMBL molecule below. Molecule depictions created with MONA [63, 66]. *This figure was originally published in [1].*

Microtubule-Destabilizing Molecules

The second experiment in this section demonstrates how the enumeration algorithm can be used to generate a diverse set of molecules with a distinct functional group. In this case, the fragment of interest only contains a single open linker to which other fragments can be attached. The target that is investigated is a heterodimer consisting of α- and β-tubulin (PDB 4O2A). These dimers assemble into larger structures, called microtubules, which are a component of the cytoskeleton of a cell. The cytoskeleton is responsible for the shape of a cell, transport of material within a cell, and coordination of cell division. The involvement in these cellular functions makes these proteins an interesting target for cancer research. Therefore, molecules are designed that prevent the formation of these dimers. In a recent publication, La Regina et al. synthesized and tested a number of microtubule-destabilizing anticancer agents [136]. All molecules exhibit a distinct trimethoxyphenyl moiety, which will be used during enumeration (see Table 8.1).

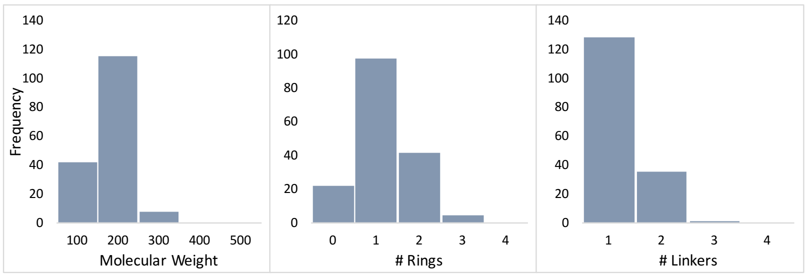


Figure 8.14: Property distributions of the fragment space constructed from Microtubule-Destabilizing molecules.

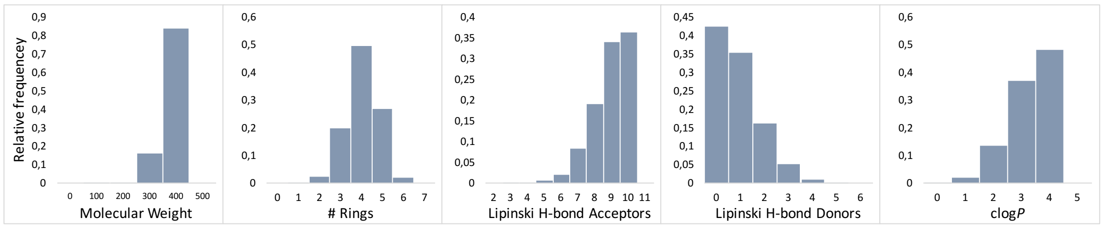


Figure 8.15: Property distributions of the enumerated molecules with trimethoxyphenyl moiety.

ChEMBL was queried with PDB 4O2A, which La Regina et al. used for their docking studies. This led to ChEMBL target ID CHEMBL3394 for which 360 unique bioactive compounds could be downloaded. 75 molecules could not be

Table 8.3: Number of enumerated potential microtubule-destabilizing molecules for high similarity ranges.

≥ 0.6	≥ 0.6	≥ 0.7	≥ 0.8	≥ 0.9	$= 1.0$
37,488	6624	995	136	67	30

fragmented with BRICS rules. The resulting fragment space contains 165 fragments, three of which contain the trimethoxyphenyl pattern (F58, F61, F64) and are shown in Table 8.1. They are used as starting points for the enumeration. For this experiment, plain drug-like constraints were used (see section 7.3) and during enumeration, a total of 575,864 molecules were generated. These molecules were compared to the initial ChEMBL actives by ECFP-Tanimoto similarity as shown in Figure 8.16. 37,488 molecules have a Tanimoto similarity greater than or equal to 0.5, see Table 8.3 for a detailed breakdown. As with the previous experiment, the molecules with the highest similarity values are depicted in Figure 8.13 c together with the corresponding active molecules.

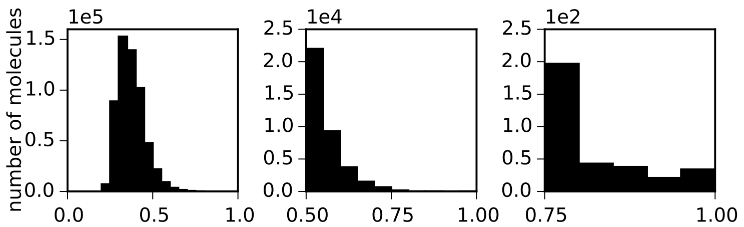


Figure 8.16: Similarity distribution of enumerated molecules with trimethoxyphenyl moiety compared to bioactive α- and β-tubulin binders from ChEMBL. Similarity range from 0.0 to 1.0 is shown on the left, 0.5 to 1.0 in the middle, and 0.75 to 1.0 on the right.

8.2.5 Enumeration of a Large Lead-Like Library

In the previous experiments, target specific fragment spaces were enumerated that were composed of fragments from already known inhibitors, whereas in this experiment, a diverse set of molecules is used to create a fragment space. This fragment space is enumerated with lead-like$_Z$ properties (see section 7.3) in order to generate a large library of molecules that can serve as a starting point for idea generation, HELLS.

In order to construct the fragment space, the "Approved Drugs" set was downloaded from DrugBank[13] [84]. 1537 molecules were then fragmented according to BRICS rules [34]. 528 molecules could not be processed since none of the BRICS rules could be applied. The remaining 1009 molecules were fragmented into 1214 fragments. An overview of the properties of these fragments is provided in Figure 8.17. Since all fragments are directly derived from actual drug molecules, the resulting enumerated molecules will consequently be composed of patterns that have previously been observed in approved agents. Therefore, HELLS may contain a lot of new, promising molecules.

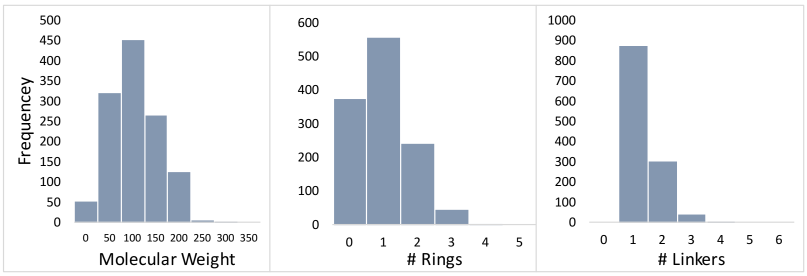


Figure 8.17: Property distribution of the fragments from the "Approved Drugs" fragment space.

The enumeration of a fragment space of this size would result in an extremely long runtime. In order to enumerate this space in a reasonable amount of time, an alternative strategy is used: Instead of a single long running enumeration, the problem is partitioned, so that several shorter running enumerations are carried out. By starting one process for each (initial) fragment on a computer cluster, the enumeration is parallelized. In oder to further limit the search space to interesting and diverse structures, the starting fragments were selected accordingly. Only fragments with at least two linkers and at least one ring of five or more atoms were considered. 183 of the 1214 fragments fulfill these criteria, thus 183 individual enumerations were performed. In this experiment, the runtime was not limited.

In an accumulated computing time of 1441 CPU hours more than 820 Million molecules were generated. The individual enumeration times and number of molecules are listed in Appendix B.3. Since all enumerations were carried out independently, the same molecule may have been generated by separate processes. In order to eliminate redundant molecules, all molecules need to be merged and filtered for duplicates. This was done based on their unique SMILES representation [114]. The SMILES strings were merged into one file and then sorted and made unique on the command line with an invocation of the GNU command *sort*[14]. The initial set of 820,467,614 molecules was reduced to 503,974,653 unique

[13]Version 4.2, see section 7.1
[14]http://www.gnu.org

molecules. Figure 8.18 shows an overview of the properties of the enumerated molecules. The Hamburg enumerated lead-like set was made available free of charge [1].

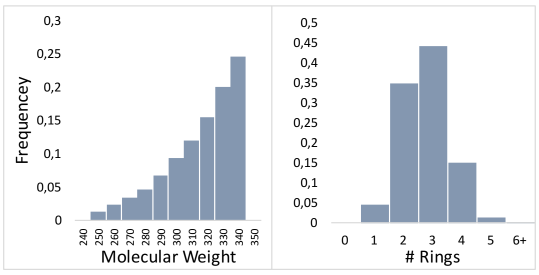


Figure 8.18: Property distributions of the HELLS library.

Enumerated molecules were compared to actual molecules from ZINC [131] and SureChEMBL [128] based on unique SMILES. Therefore, both sets were converted into unique SMILES with the NAOMI converter [59] thus filtering out erroneous molecules. The "lead-like" set of ZINC contains 7,096,164[15] and SureChEMBL 12,790,627 molecules[16]. HELLS contains 48,814 molecules from ZINC and 21,825 molecules from SureChEMBL. In other words, 0.0097% of HELLS were found in ZINC, 0.0043% in SureChEMBL. Although neither collection exhaustively covers the known chemical universe, ZINC as a big resource of purchasable compounds and SureChEMBL as a resource of patented molecules represent good approximations. This becomes even clearer when comparing ZINC to the most exhaustive resource to date, i.e., the CAS registry of the Chemical Abstract Service [52]. It contains "more than 109 million unique organic and inorganic chemical substances" [52]. Considering that it also contains a lot of inorganic compounds and ZINC contains over 35 million organic compounds [131], the coverage of ZINC is quite impressive.

8.3 Reaction-based Fragment Spaces

This section discusses experiments that utilize the algorithm for the construction of reaction-based fragment spaces, which is described in section 6.2. The input for the method is a set of molecules and a set of reactions. The fragment spaces constructed in this way have a high likelihood of generating synthesizable molecules when subject to one of the existing methods [24, 25, 27, 29, 31, 32] or the FSees algorithm.

[15]Accessed on September 15th, 2015
[16]Accessed on January 27th, 2016

The ChemBridge Building Blocks library (see section 7.1.6) that was downloaded from ZINC [131] serves as source for input molecules. It contains purchasable building blocks that are typically used to synthesize larger molecules. The set contains 17,974 unique molecules that have molecular properties much lower than drug-like molecules (see Figure 8.19). Reactions were extracted from the supplemental material of Hartenfeller et al. [96] (see 7.2).

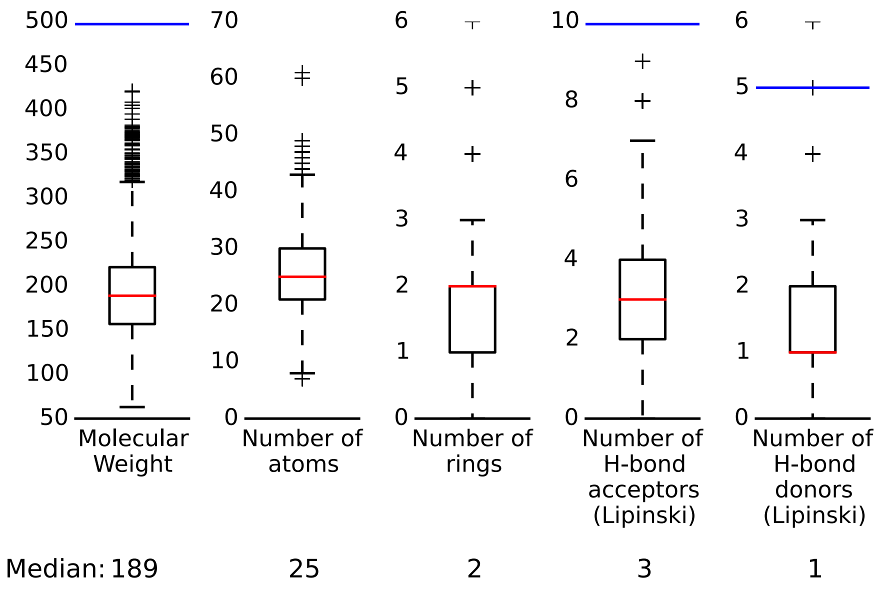


Figure 8.19: Properties of the ChemBridge Building Blocks dataset. Red lines denote the median, blue lines the upper bound of the drug-like constraints for applicable properties.

8.3.1 Preparation of Reactions

The reaction set provided by Hartenfeller et al. contains a total of 58 reactions specifically compiled for use in a computational context. Although these reactions can be read, they cannot be applied directly due to conceptual challenges of fragment spaces (see section 6.2). 29 of these reactions constitute ring forming reactions that can currently not be modeled. These reactions were removed. The remaining 29 reactions introduce one single or double bond between two compatible fragments. Their names are listed in Table 8.4, the corresponding SMIRKS in appendix C.3. During tests, it became apparent that a few reactions could not be processed with the proof of concept implementation in the current state. These unsuitable reactions were not considered for the experiments below. The reason for their failure is discussed on a case-by-case basis in the following.

Table 8.4: List of non ring-forming reactions from the set of Hartenfeller et al. [96]. Reactions marked with an asterisk could not be processed and are discussed in the text.

Buchwald-Hartwig	Negishi
decarboxylative coupling*	nucl sub aromatic ortho nitro
Grignard alcohol*	nucl sub aromatic para nitro
Grignard carbonyl*	piperidine indole
Heck non-terminal vinyl	reductive amination
Heck terminal vinyl	Schotten-Baumann amide
heteroaromatic nuc sub	Sonogashira
Mitsunobu imide	Stille
Mitsunobu phenole	sulfon amide
Mitsunobu sulfonamide	Suzuki
Mitsunobu tetrazole 1	thiourea*
Mitsunobu tetrazole 2*	urea*
Mitsunobu tetrazole 3*	Williamson ether
Mitsunobu tetrazole 4	Wittig
N-arylation heterocycles	

In the case of *decarboxylative coupling*, one of the reactants cannot be matched to any molecule in the building blocks set. Although the other reactant is found 1573 times, it does not make sense to construct a fragment space with this reaction since no connections can be made.

Reactions *Mitsunobu tetrazole 2* and *Mitsunobu tetrazole 3* are identical to *Mitsunobu tetrazole 4* and *Mitsunobu tetrazole 1*, respectively, except for the position where the hydrogen is split off. At the moment, the algorithm only splits off hydrogens from the atom at which the linker is attached, i.e., where the reaction takes place. Because all four reactions match the same molecules and reactions 2 and 4 as well as 1 and 3 result in the same product, cases 2 and 3 can be neglected due to redundancy.

The following four reactions – as depicted in Figure 8.20 – were excluded because the bond changes that are introduced are rather complex or do not take place at the reactive atom, as described in section 6.2.3. Reactions *urea* and *thiourea* are identical, except that an oxygen atom is present and in the latter a sulfur atom. In both cases, a double bond to a neighboring atom must be broken up and the neighboring atom must subsequently be saturated with a hydrogen. Since this change does not take place at the reactive atom, it is currently not modeled. In the *Grignard reaction* alkyl-, vinyl-, or aryl-magnesium halides react with electrophilic groups. In the case of *Grignard alcohol* the situation is identical to the previous case: a double bond between the reactive atom and a neighboring oxygen is removed. The case of *Grignard carbonyl* is even more complex; here,

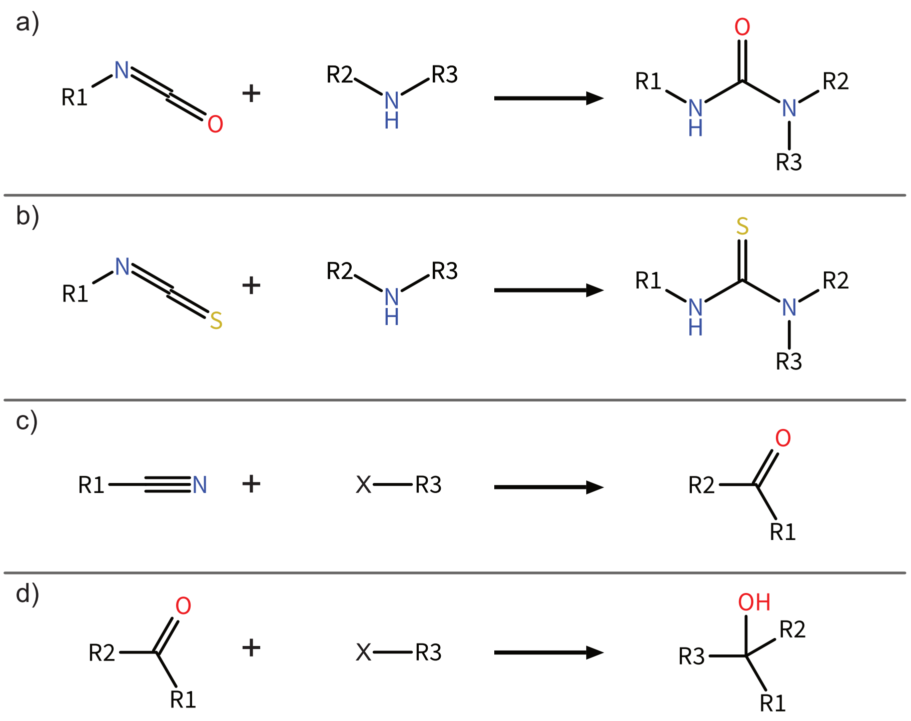


Figure 8.20: Four reactions that cannot be modeled, currently: a) Urea b) Thiourea c) Grignard carbonyl d) Grignard alcohol. See text for details. In this case, R1-R3 denote parts that differ between molecules, not linkers.

the reaction takes place at a nitrile group. The triple bond is removed and the nitrogen is split off and substituted by a double bonded oxygen. The latter only occurs in the product and must therefore be created and added to the reactive atom.

8.3.2 Fragment Space Construction

For each of the 22 reactions that can be modeled, a fragment space was created. Since only one reaction was used, such a fragment space essentially represents a combinatorial library. The results are summarized in Table 8.5. The number of fragments per space varies from 68 to 9,280 fragments and adds up to 65,913. Most fragments have only one linker, 36 fragments feature two and only one fragment features three linkers. 26 of the fragments with more than one linker matched both reactants of the respective reactions (Buchwald-Hartwig and Negishi). Several reactions show the same number of fragments for one of the link types. In these cases, the functional groups are identical, although the second reactant is always different (e.g. Heck non-terminal vinyl and Heck terminal

Table 8.5: Composition of fragment spaces for individual reactions as number of fragments. Link types L1 and L2 correspond to the first and second reactant of the corresponding reaction.

Reaction	Total	No. of linkers			Link type		
		1	2	3	L1	L2	both
Buchwald-Hartwig	7028	7014	13	1	5635	1383	10
Heck non-terminal vinyl	2029	2029	0	0	2017	12	0
Heck terminal vinyl	2019	2019	0	0	2017	2	0
Mitsunobu imide	863	863	0	0	27	836	0
Mitsunobu phenole	1187	1187	0	0	352	835	0
Mitsunobu sulfonamide	885	885	0	0	50	835	0
Mitsunobu tetrazole 1	858	858	0	0	23	835	0
Mitsunobu tetrazole 4	843	843	0	0	8	835	0
N-arylation heterocycles	994	994	0	0	835	159	0
Negishi	5413	5397	16	0	2789	2608	16
Schotten-Baumann amide	9280	9277	3	0	6676	2604	0
Sonogashira	888	888	0	0	34	854	0
Stille	2972	2972	0	0	2165	807	0
Suzuki	1764	1764	0	0	1618	146	0
Williamson ether	2027	2027	0	0	593	1434	0
Wittig	2294	2292	2	0	593	1701	0
heteroaromatic nuc sub	4766	4766	0	0	4493	273	0
nucl sub aromatic ortho nitro	4503	4503	0	0	4493	10	0
nucl sub aromatic para nitro	4500	4500	0	0	4493	7	0
piperidine indole	68	68	0	0	42	26	0
reductive	6195	6193	2	0	4456	1739	0
sulfon-amide	4537	4537	0	0	4493	44	0
All	65913	65876	36	1	47902	17985	26

vinyl). This analysis quantifies how many fragments are essentially identical and differ only in the link type at the reactive site.

Then, all reactions were used together in order to create a multi-reaction fragment space. The way reactions and molecules are processed by the algorithm leads to a dependence on the order of reactions. Reactions are iterated in an outer loop, while molecules and – previously generated – fragments are iterated repeatedly for each reaction. In the case of a molecule that has functional groups for two reactions, linkers are added for the first reaction resulting in a fragment. This fragment is subject to matching of the second reaction. When the earlier reaction removed an atom that is relevant for the second reaction, the second reaction cannot be matched to the fragment. If the order were reversed, this fragment may have been matched twice. This is a drawback of the implemented matching strategy II as described in section 6.2.3. A solution to this problem is to run the algorithm with all possible reaction orders and merge the resulting fragments into a unique set; or to apply a possibly modified version of strategy I (see section 6.2.3). In the following, four reaction orders were used to show the variability caused by this.

Two strategies regarding assignment of the link types were used. The naive approach uses different link types for each reaction (*reaction-level*), the second the same link types for the same functional groups (*functional group-level*). The latter describes a more realistic scenario since functional groups do take part in different reactions. For example, in the case of all *Mitsonobu* reactions, one of the reactants is always a ketone, as described by the following SMARTS: `[C;H1&$(C([#6])[#6]),H2&$(C[#6]):1][OH1]`. In the first case, a different link type is assigned for each reaction, in the second case, the same link typ is always assigned, i.e., L17. Applying the different strategies to all reactions, the resulting fragment space contains 44 link types in the first case and only 34 in the second. More importantly, in the second (functional group-level) case, the number of fragments is reduced significantly (see Table 8.6). A molecule that can take part in multiple reactions is represented by a single instance rather than by multiple copies that are structurally identical but only differ in the link type. This "reuse" of a fragment is one of the major advantages of a fragment space since it allows to represent a large amount of molecules in a compact way.

Table 8.6 shows the results from these construction experiments. Each of the two strategies was applied four times with different input regarding the order of reactions. The average number of fragments is 201,906 for the reaction-level strategy and 102,746 for the functional group-level strategy. Applying the more realistic scenario hence results in a reduction of fragments by almost 50%. The reduced number of fragments in the case of functional group-level link types shows that the previous set contained many redundant fragments. This effect increases with the number of linkers. When comparing the generation of individual reaction fragment spaces with multiple reaction fragment spaces, it can be seen that the number of molecules with one linker is identical in the case of reaction-level link types, i.e., 65,876 (see Tables 8.5 and 8.6). However, the number of fragments with multiple linkers is significantly higher than the initial 36 and the maximal number of linkers increases from three to seven. This indicates that many of the molecules in the input set can be subject to multiple reactions. This behavior was expected for a library that was designed as building blocks for molecular synthesis. The reduced number of fragments in the case of functional group-level link types shows that the previous set contained many redundant fragments. This effect is increasing with the number of linkers as shown by the percentage in the last row of Table 8.6.

Table 8.6: Composition of fragment spaces for all *working* reactions from the Hartenfeller set [96] as number of fragments. Letters A-H denote that the reaction input was differently sorted. The percentages in the last row refer to how many molecules from the reaction-level set are in the functional group-level set. The percentage was computed from mean values.

Link Type Set	Order	Total	No. of linkers						
			1	2	3	4	5	6	7
reaction-level	A	208194	65876	89165	41029	8413	1775	1395	540
	B	196382	65876	82148	36911	7890	1721	1325	510
	C	196247	65876	83161	36771	7272	1481	1220	465
	D	206802	65876	88399	41184	8161	1496	1220	465
	Mean	201906	65876	85718	38974	7934	1618	1290	495
functional group-level	E	101867	46235	40145	13101	2076	155	108	46
	F	104087	46235	41487	13528	2252	264	224	96
	G	98970	46235	38246	12245	1931	164	104	44
	H	106059	46235	42048	14480	2609	318	256	112
	Mean	102746	46235	40482	13339	2217	225	173	75
		51%	70%	47%	34%	28%	14%	13%	15%

From the 22 working reactions, several fragment spaces with an average number of 102,746 molecules were successfully constructed. More than half of these molecules exhibit more than one linker and many of these may partake in multiple reactions. Therefore, molecules can be constructed from more than one fragments in contrast to combinatorial libraries, where only pairwise reactions are modeled. The fragment spaces constructed here are of great value since they are composed of building blocks and concrete synthetic reactions. Every molecule retrieved from this space comes with a list of building blocks and the reactions to connect them so that the transition to the laboratory is facilitated.

When working with these spaces to construct molecules, several aspects have to be considered. Depending on the desired search algorithm, it may be required to reduce the size of the fragment space. For a similarity search with FTrees-FS [24] or a substructure search [32, 33], this space is feasible, for enumeration it is not. In section 8.2, several spaces with less than 500 fragments were described that could not be enumerated. To reduce the size, filtering rules could be applied to the input, during the construction, or in the fragment space itself. The size of the fragment space can be influenced by using a smaller input library or a smaller

Table 8.7: Computing times for the construction of fragment spaces with functional group-level link types (see Table 8.6).

Order Set	Time [hh:mm:ss]
E	00:40:38
F	00:46:44
G	00:51:46
H	00:56:40

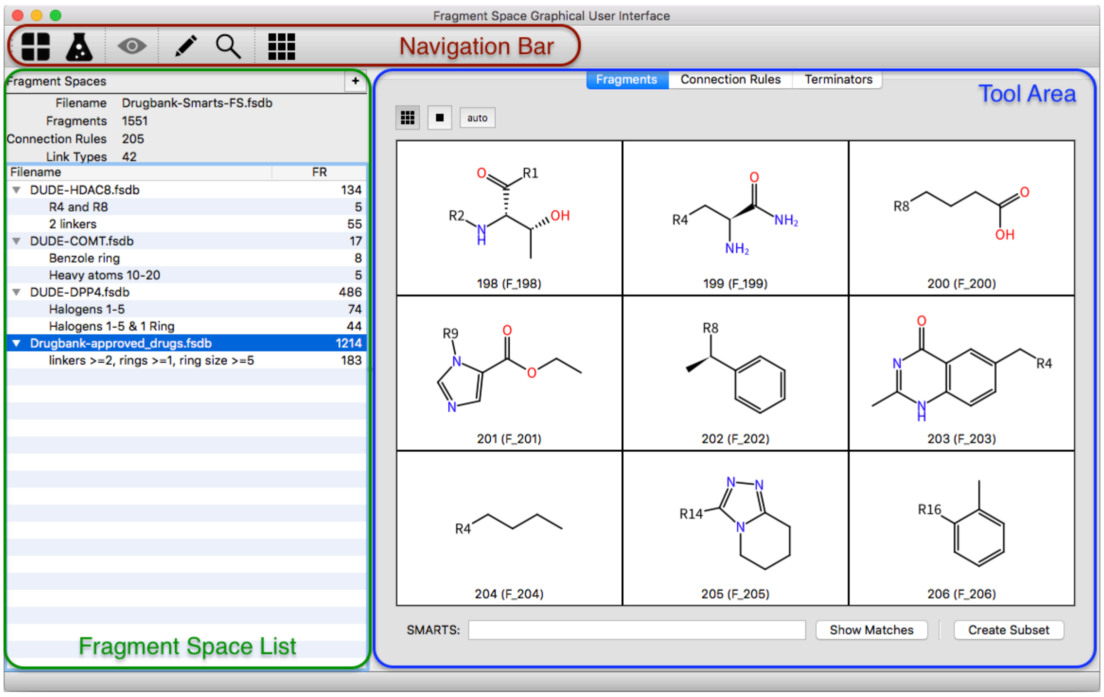
number of reactions, e.g., reactions that are often consecutively used in organic synthesis. In this way, a number of fragment spaces could be constructed, e.g., for often used synthesis protocols. During construction, an automated filtering step could be added to sort out fragments with impractical properties. For instance, fragments with seven linkers may partake in up to seven different reactions, which is not a manageable number. All these modifications may also be applied on the fragment space data structure after successful construction. The tool to do this is Fragment Space Commander and will be introduced next in chapter 9.

9 Software and Tools

During the course of this project, a number of tools and NAOMI library functions were implemented to work with fragment spaces and molecules. This includes several command line tools that provide interfaces to the newly developed and existing algorithms, i.e., fragment space enumeration (FSees, section 9.2), fragment space construction based on reactions (section 9.3) and retrosynthetic fragmentation (section 9.4), fingerprint similarity (section 9.5), as well as the graphical user interface Fragment Space Commander that enables to interactively work with fragment spaces (section 9.1). Furthermore, the functionality of the NAOMI library was extended by a fragment space file format. This chapter covers the additions made to the NAOMI library and the tools implemented for this project. A special focus is put on Fragment Space Commander, which represents one of the major efforts of this project and took up a large part of the software development time.

9.1 Fragment Space Commander

Software usability is an important aspect for bio- and cheminformatics tools. Many methods are quite complex and have many parameters. Providing easily accessible tools with sensible default values is key to the success of a method. In addition, related methods are not always developed and published together. However, sometimes it would make sense to use them together. There are work-



Filename	Drugbank-Smarts-FS.fsdb
Fragments	1551
Connection Rules	205
Link Types	42

Filename	FR
▼ DUDE-HDAC8.fsdb	134
R4 and R8	5
2 linkers	55
▼ DUDE-COMT.fsdb	17
Benzole ring	8
Heavy atoms 10-20	5
▼ DUDE-DPP4.fsdb	486
Halogens 1-5	74
Halogens 1-5 & 1 Ring	44
▼ Drugbank-approved_drugs.fsdb	1214
linkers >=2, rings >=1, ring size >=5	183

Figure 9.1: Window of Fragment Space Commander with highlighted components. The Navigation Bar provides a list of available tools shown in Figure 9.2. The Fragment Space List on the left shows loaded fragment spaces and available fragment sets. The Tool Area is variable and changes depending on the selected tool. Here, fragments of the "Drugbank Approved Drugs" fragment space are shown (see section 8.2.5).

flow tools such as Knime[17], Pipeline Pilot[18], or Orange[19] to connect individual tools that are available as executables and to carry out powerful data analysis. They are convenient when designing a workflow that is used very often and modified seldom. However, they are not very convenient for interactive work or to quickly "try something out". Graphical user interfaces (GUIs) fill this gap and allow interested users to both quickly evaluate a method and use several related methods conveniently together. They provide convenient access to cheminformatics algorithms and – at the same time – enable the visualization of parameters, both input and output. Especially for scientists that are not programmers, casual users, or reviewers who want to quickly evaluate a method, such a graphical interface is valuable.

Within the scope of this project a tool for working with fragment spaces was developed. Fragment Space Commander (FSC) provides access to the newly

[17]http://www.knime.org
[18]http://accelrys.com
[19]http://orange.biolab.si

developed methods for reaction-based construction and enumeration as well as previously published fragment space methods that were to date only accessible via command line tools [24, 31, 32, 34]. It allows to create, edit, and query fragment spaces in a number of ways. Figure 9.1 shows the FSC window and its components highlighted. The hierarchy of the different tools and related views is shown in Figure 9.2. This section features several screenshots of FSC showing different components of interface. A complete list of all views is shown in Appendix A.1.

For the visualization of fragments and molecules, FSC uses NAOMI functionality, which was developed by Matthias Hilbig for use in MONA [63, 66].

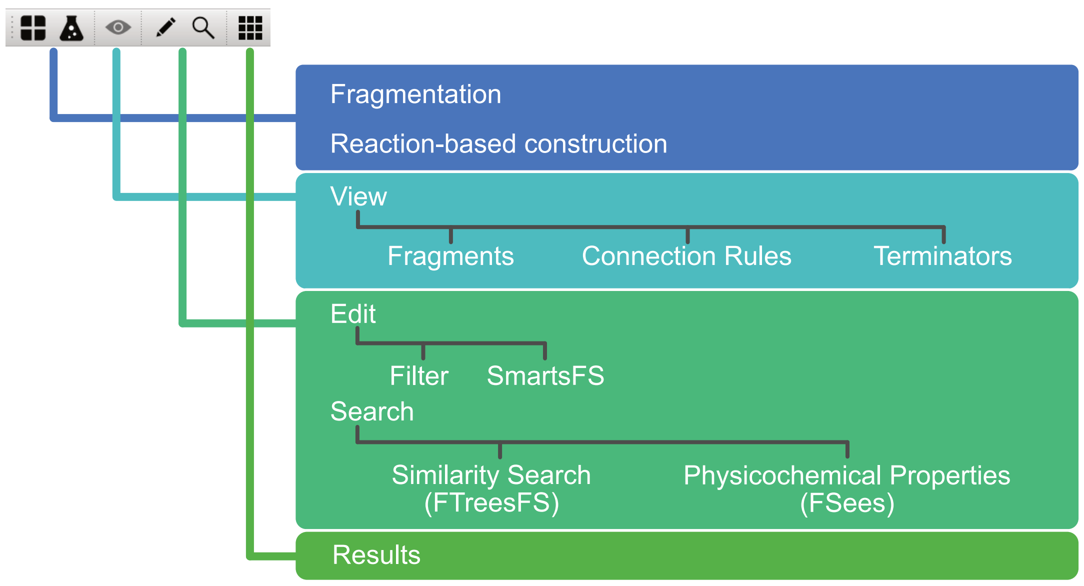

Fragmentation
Reaction-based construction
View
Fragments
Connection Rules
Terminators
Edit
Filter
SmartsFS
Search
Similarity Search (FTreesFS)
Physicochemical Properties (FSees)
Results

Figure 9.2: Hierarchy of the tools and views of Fragment Space Commander.

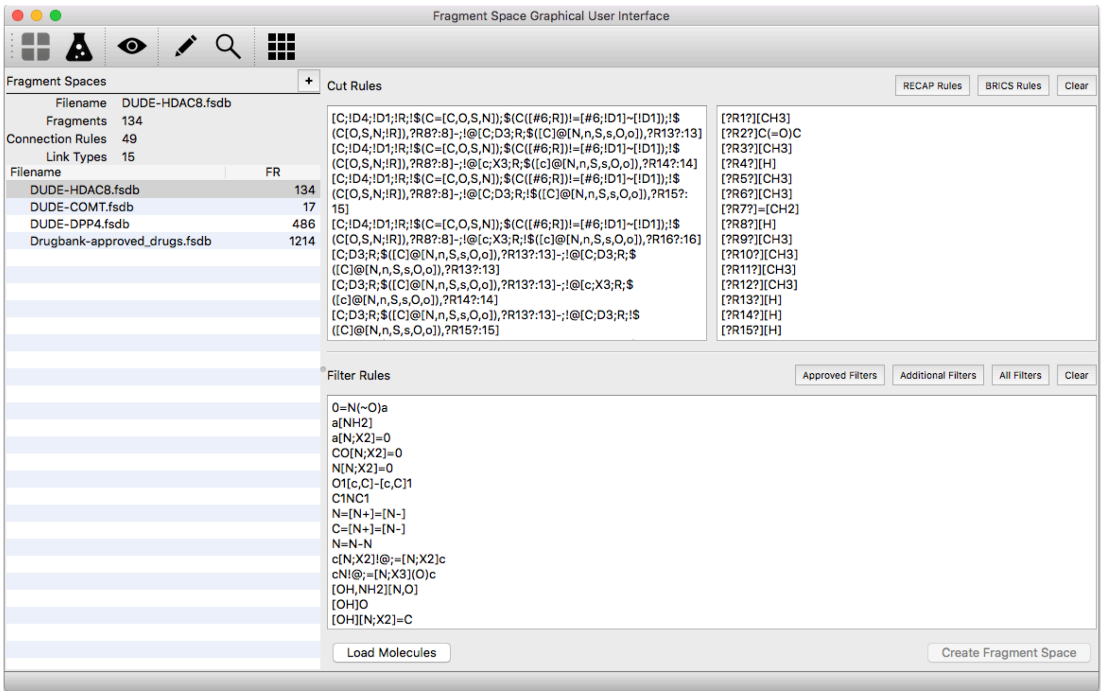
9.1.1 Creation

To create fragment spaces, the tool provides access to the fragmentation approach described in section 6.1 as well as the reaction-based approach in section 6.2. Molecules can be provided as SMILES, mol2, or SDF files.

The fragmentation procedure requires retrosynthetic cut rules to be supplied as SMARTS pattern [100] and terminal fragments as SMILES strings [113]. If no terminators are provided, default terminators will be used, i.e., hydrogen for a single bond, CH_2 for double bonds, and CH for triple bonds. Both modified RECAP and BRICS rules are available as pre-defined sets and can be accessed via a single click. In addition, a list of SMARTS pattern used to filter out unwanted fragments can be supplied. This post-filtering is also used by the BRICS method

[34]. In BRICS, the toxicity filters by Kazius et al. were used. FSC provides one-click access to two sets of toxicophores (*approved* and *all*) [120].

The reaction-based construction method requires a list of SMIRKS [100] each with a reaction name and the link types associated with both reactants as input. No other data is required.



Filename	DUDE-HDAC8.fsdb
Fragments	134
Connection Rules	49
Link Types	15

Filename	FR
DUDE-HDAC8.fsdb	134
DUDE-COMT.fsdb	17
DUDE-DPP4.fsdb	486
Drugbank-approved_drugs.fsdb	1214

```
[C;!D4;!D1;!R;!$(C=[C,O,S,N]);$(C([#6;R])!=[#6;!D1]~[!D1]);!$
(C[O,S,N;!R]),?R8?:8]-;!@[C;D3;R;$([C]@[N,n,S,s,O,o]),?R13?:13]
[C;!D4;!D1;!R;!$(C=[C,O,S,N]);$(C([#6;R])!=[#6;!D1]~[!D1]);!$
(C[O,S,N;!R]),?R8?:8]-;!@[c;X3;R;$([c]@[N,n,S,s,O,o]),?R14?:14]
[C;!D4;!D1;!R;!$(C=[C,O,S,N]);$(C([#6;R])!=[#6;!D1]~[!D1]);!$
(C[O,S,N;!R]),?R8?:8]-;!@[C;D3;R;!$([C]@[N,n,S,s,O,o]),?R15?:
15]
[C;!D4;!D1;!R;!$(C=[C,O,S,N]);$(C([#6;R])!=[#6;!D1]~[!D1]);!$
(C[O,S,N;!R]),?R8?:8]-;!@[c;X3;R;!$([c]@[N,n,S,s,O,o]),?R16?:16]
[C;D3;R;$([C]@[N,n,S,s,O,o]),?R13?:13]-;!@[C;D3;R;$
([C]@[N,n,S,s,O,o]),?R13?:13]
[C;D3;R;$([C]@[N,n,S,s,O,o]),?R13?:13]-;!@[c;X3;R;$
([c]@[N,n,S,s,O,o]),?R14?:14]
[C;D3;R;$([C]@[N,n,S,s,O,o]),?R13?:13]-;!@[C;D3;R;!$
([C]@[N,n,S,s,O,o]),?R15?:15]
```

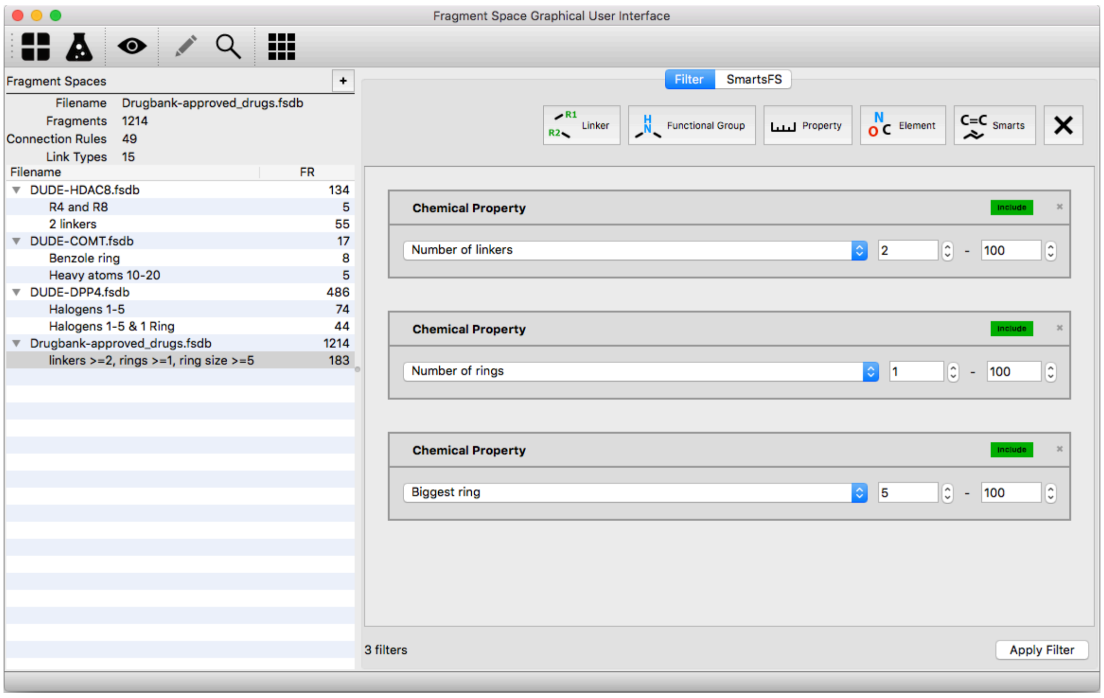
```
[?R1?][CH3]
[?R2?]C(=O)C
[?R3?][CH3]
[?R4?][H]
[?R5?][CH3]
[?R6?][CH3]
[?R7?]=[CH2]
[?R8?][H]
[?R9?][CH3]
[?R10?][CH3]
[?R11?][CH3]
[?R12?][CH3]
[?R13?][H]
[?R14?][H]
[?R15?][H]
```

```
0=N(~O)a
a[NH2]
a[N;X2]=0
CO[N;X2]=0
N[N;X2]=0
O1[c,C]-[c,C]1
C1NC1
N=[N+]=[N-]
C=[N+]=[N-]
N=N-N
c[N;X2]!@;=[N;X2]c
cN!@;=[N;X3](O)c
[OH,NH2][N,O]
[OH]O
[OH][N;X2]=C
```

Figure 9.3: Fragmentation view of Fragment Space Commander. Three entry fields are displayed with data for a fragmentation based on BRICS [34], cut rules (top left), terminators (top right), and filter rules [120] (bottom). Four fragment spaces are open and shown in the fragment space list.

9.1.2 Modification

Modification of fragment spaces is possible for the first time with a graphical tool. Before, fragment space files had to be modified by hand. This functionality allows to create tailored spaces to which search algorithms can be applied more efficiently since the search space is smaller. Two strategies for filtering fragments are available: The first is based on molecular properties and it allows the user to filter fragments according to physicochemical properties, functional groups, and linker types. Filtered sets can be used to create new fragment spaces. The second strategy is based on substructure searching and requires a user defined SMARTS pattern. SmartsFS was developed and published by Ehrlich et al. [32]. The algorithm finds all fragments and fragment combinations containing the pattern and creates a new fragment space from them.



Filename	Drugbank-approved_drugs.fsdb
Fragments	1214
Connection Rules	49
Link Types	15

Filename	FR
DUDE-HDAC8.fsdb	134
R4 and R8	5
2 linkers	55
DUDE-COMT.fsdb	17
Benzole ring	8
Heavy atoms 10-20	5
DUDE-DPP4.fsdb	486
Halogens 1-5	74
Halogens 1-5 & 1 Ring	44
Drugbank-approved_drugs.fsdb	1214
linkers >=2, rings >=1, ring size >=5	183

Chemical Property include
Number of linkers 2 - 100

Chemical Property include
Number of rings 1 - 100

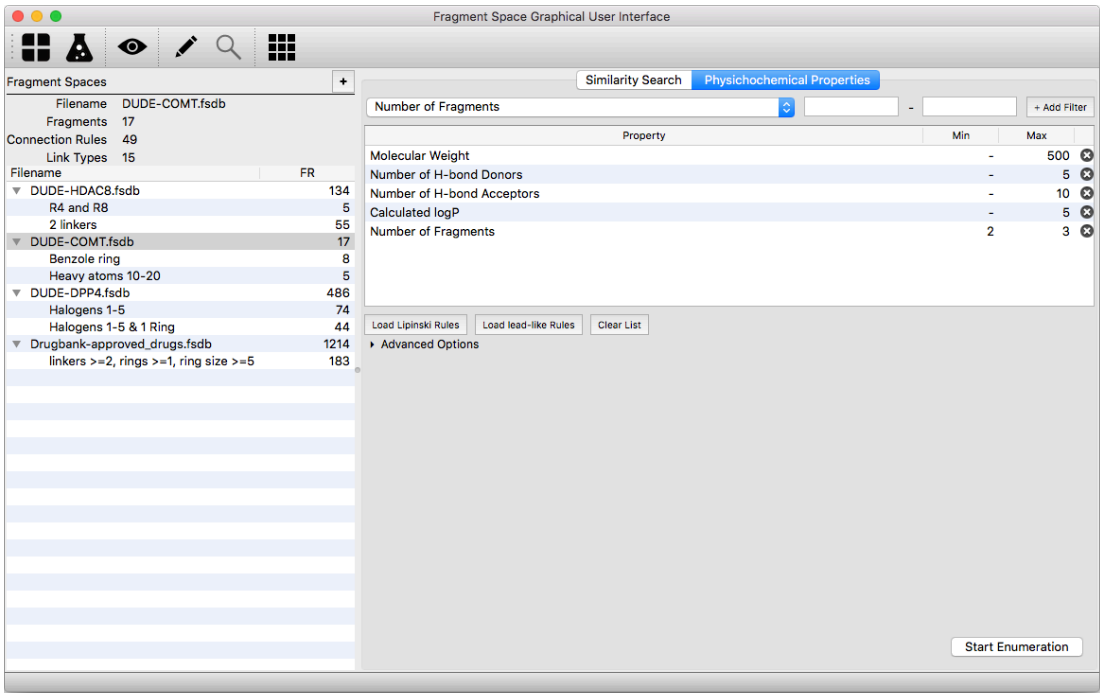
Chemical Property include
Biggest ring 5 - 100

3 filters
Apply Filter

Figure 9.4: Filter tool view of Fragment Space Commander. Four fragment spaces respective fragment sets are displayed in the fragment space list. The tool area shows the filter view with the three filters used for the enumeration of the HELL set (see section 8.2.5).

9.1.3 Search

FSC provides access to two algorithms to retrieve molecules from a given fragment space. This is the enumeration algorithm discussed in section 6.4 and the FTreesFS similarity search algorithm [24]. The latter requires a query molecule as input, as well as a similarity cutoff and it deterministically generates a ranked list of molecules that are similar to a query molecule. Similarity is assessed based on the feature tree descriptor [58], which represents a molecule as a tree of fragments. It has been developed in the fragment space context and is therefore tailored to the problem.


Fragment Space Graphical User Interface

Fragment Spaces

Filename	DUDE-COMT.fsdb
Fragments	17
Connection Rules	49
Link Types	15

Filename	FR
▼ DUDE-HDAC8.fsdb	134
R4 and R8	5
2 linkers	55
▼ DUDE-COMT.fsdb	17
Benzole ring	8
Heavy atoms 10-20	5
▼ DUDE-DPP4.fsdb	486
Halogens 1-5	74
Halogens 1-5 & 1 Ring	44
▼ Drugbank-approved_drugs.fsdb	1214
linkers >=2, rings >=1, ring size >=5	183

Similarity Search | Physicochemical Properties

Number of Fragments - + Add Filter

Property	Min	Max
Molecular Weight	-	500
Number of H-bond Donors	-	5
Number of H-bond Acceptors	-	10
Calculated logP	-	5
Number of Fragments	2	3

Figure 9.5: Enumeration view of Fragment Space Commander. Four fragment spaces respective fragment sets are displayed in the fragment space list. The tool area shows the parameters for an enumeration with drug-like properties based on the Rule of five.

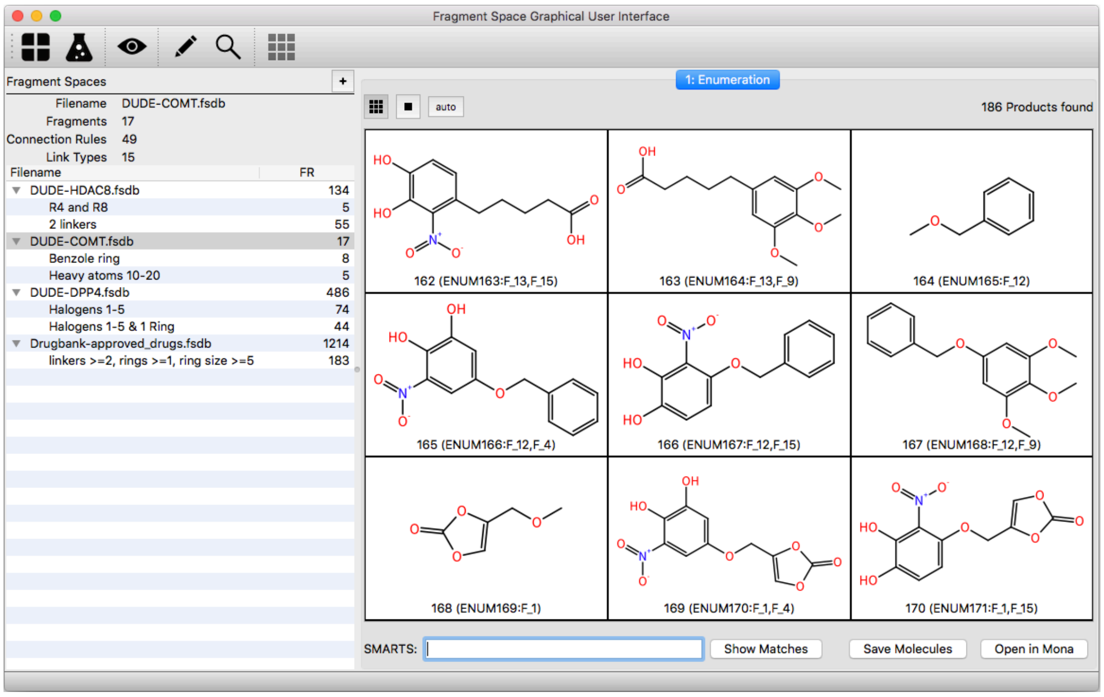
9.1.4 Workflow

Fragment Space Commander unites these methods into a single user interface. Fragment spaces cannot only be created, but they can also be loaded from Fragment Space Format (FSF) and Fragment Space Database (FSDB) files, as well as saved to FSDB files (see 9.1.5). Once a fragment space is loaded, the different methods can be used independently or in an interactive workflow.

An example of a workflow to recreate the enumeration experiment for a DUD-E target (see 8.2.3) is as follows:

1. Select the *Fragmentation View* and load the file with *known actives*.
2. Click on the BRICS button to load the BRICS rules.
3. Click on the *all toxicophores* button to load the post processing rules.
4. Switch to the *Query View* and select the *Physicochemical Properties* Tab.
5. Enter the enumeration constraints from Table B.2 in Appendix B.
6. Click *Enumerate* and wait until the enumeration finishes[20].
7. Switch to the Results View to view the resulting molecules.

[20]Note that enumerations for some targets may have a very long runtime.



Filename	DUDE-COMT.fsdb
Fragments	17
Connection Rules	49
Link Types	15

Filename	FR
DUDE-HDAC8.fsdb	134
R4 and R8	5
2 linkers	55
DUDE-COMT.fsdb	17
Benzole ring	8
Heavy atoms 10-20	5
DUDE-DPP4.fsdb	486
Halogens 1-5	74
Halogens 1-5 & 1 Ring	44
Drugbank-approved_drugs.fsdb	1214
linkers >=2, rings >=1, ring size >=5	183

Figure 9.6: Result view of Fragment Space Commander. Enumeration results for the enumeration of the DUD-E COMT fragment space with drug-like properties as depicted in Figure 9.5.

A second example describes the creation of a tailored fragment space and a similarity search in this space:

1. Click the + button in the *Fragment Spaces List View* on the left to load a fragment space.
2. Select the *Filter* tab in the *Edit View*.
3. Click any of the filter buttons to add filters as desired.
4. Click *Apply Filter*.
5. Select the newly created fragment set in the *Fragment Spaces List View*.
6. Right Click and select *New Fragment Space From Set*.
7. Select the newly created Fragment Space.
8. Switch to the *Query View* and select the *Similarity Search* tab.
9. Enter a SMILES string as query molecule.
10. Click *Search* and wait until the search has finished.
11. Switch to the *Results View* tab to view the resulting molecules.

9.1.5 Fragment Space Storage

In conjunction with the development of Fragment Space Commander a new way of storing fragment spaces was added. Instead of using the existing text-based format, for which no writing capabilities exist in the NAOMI library, a new SQLite based format was developed. It uses database functionality from the NAOMI library, which was contributed by Therese Inhester and is also used in MONA [63, 66]. The SQLite based format has the advantage that additional tables can be added easily, e.g., for storing fragment sets, and the information in the database can be accessed conveniently, e.g., for filtering of fragments. This new format is the basis for Fragment Space Commander (FSC).

9.1.6 Fragment Space Format

A fragment space consists of fragments with annotated linkers and connections rules (see section 3.4). The fragment space file (FSF) format encodes this information in a text file (see C.1 for an example). There are four sections: *link_types*, *fragment_files*, *link_terminal_groups*, and *link_compatibility_matrix*. *link_types* simply lists all valid link names. *fragment_files* contains a list of fragment files in either SMILES, Mol2, or SDF format. *link_terminal_groups* contains a lists of terminal fragments, one for each link type. This can either be a SMILES string or a filename. *link_compatibility_matrix* contains a lists of all compatible pairs of link types. Both *link_terminal_groups* and *link_compatibility_matrix* contain information about bond distance and angles. This information is an artifact from the Flex* library and is not required by NAOMI anymore to connect fragments.

The format is text based. The advantage of this is that it can be easily read and changed by hand, whereas the disadvantage is that this process is quite error prone, because each section uses a column based format. In addition, no functionality exists in the NAOMI library to write this format. A revision of this format was not undertaken.

9.1.7 Fragment Space Database

The Fragment Space Database (FSDB) format is built based on NAOMI functionality. It uses several NAOMI libraries: *DBConnection* to create and access databases, *MoleculeDB* to store fragments; and *PropertiesDatabase* to annotate molecules. The latter enables tools to conveniently search fragments for their properties, e.g., molecular mass, number of linkers, etc. *PropertiesDatabase* also provides an interface to add custom properties. This was used to annotate fragments with their link types in order to provide a search in Fragment Space Commander based on link type. In order to store connection rules, terminal fragments, and fragment sets, custom classes were implemented. Each class represents a table in the SQLite database schema (see Table 9.1 and Figure 9.7). Connection rules are stored as pairs of compatibility link types and terminators are stored as SMILES. Fragment sets are used by the Fragment Space Commander as a means to store fragment IDs. These sets can be generated by filtering fragments based on properties or by manual selection. The tables related to fragment sets are only relevant for FSC and are ignored by other applications.

Table 9.1: Mapping between C++ classes and database tables and their origin. An asterisk '*' denotes that the information is stored in several tables.

C++	SQL	Origin
`MoleculeDatabase`	`MoleculeDB_*`	NAOMI
`PropertyDatabase`	`PropertyDB_*`	NAOMI
`ConnectionRulesDataba`	`FragspaceDB_connectionrules`	new
`TerminatorDatabase`	`FragspaceDB_terminators`	new
`FragmentSetDatabase`	`FragspaceDB_fsetfragments` `FragspaceDB_fsetnames`	new

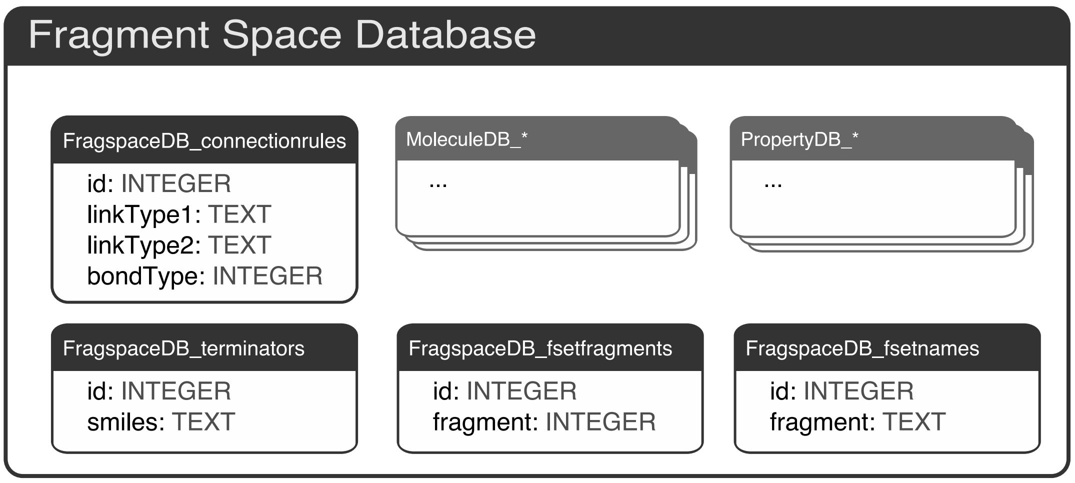


Figure 9.7: Database schema for the Fragment Space Database file format. Tables framed in dark grey were newly implemented, lighter tables are provided by the NAOMI library. An asterisk '*' denotes that the data is stored in several tables and that it is accessed via the C++ API.

9.2 FSees

The FSees command line tool implements the FSees algorithm described in section 6.4 and [1]. It can be used for long running enumerations on a local computer or on a cluster where GUI usage is not possible. The program takes two mandatory parameters, i.e., a path to a fragment space in FSF or FSDB format and a path to an output file (SMILES format). Other parameters are optional. This includes the size of the database cache and Bloom-filter in MB, a list of initial fragments for the alternative enumeration mode described in section 6.4.8, and a tolerance value, i.e., the number of properties that are allowed to be outside their thresholds. Furthermore, individual threshold ranges can be defined for each property (see Table 6.1). For a complete list of parameters, please refer to Appendix A.2. Upon execution and depending on the verbosity parameter, the FSees tool may write status information to *cout*, like fragment space information and which fragment is currently processed.

9.3 Fragspace Reaction

This command-line tool implements the algorithm for the construction of fragment spaces based on synthetic reactions as described in section 9.3. The program does not require any operational parameters, but several filenames for in- and output. As input, it requires a list of molecules (in SMILES, Mol2, or SDF format) and a list of reactions (as SMIRKS [100]). The only mandatory output parameter is a filename for the resulting fragment space, which will be written in FSDB format. Optional output includes a list of linker names with the corresponding reaction and a list of plain fragments in the NAOMI-supported molecule file formats.

9.4 Molecule Shredder

The `molecule_shredder` command-line tool provides access to the BRICS fragmentation strategy described in section 6.1 [34]. There are three types of parameters, i.e., input, output, and optional post-processing filters. Mandatory parameters are a path to a molecule file, a list of cut rules as SMARTS pattern, and a path to an output file. The output can be a list of fragments (in SMILES, Mol2, or SDF format) or a fragment space file (FSDB), or both. An optional input parameter is a list of terminators to be used in fragment space construction. The two post processing options are a flag to trigger the removal of large fragments (see section 7.2.1) and a list of exclusion rules in SMARTS pattern notation. During processing and depending on the verbosity parameter, the tool may write status information to *cout*. This is for instance the ID of the molecule being currently processed and the number of molecules that were not fragmented. For a complete list of parameters, please refer to Appendix A.4.

9.5 Fingerprint Similarity Tool

The fingerprint-similarity tool is a command line tool that was used for analysis of the enumeration results. Since computing molecule similarity is an every-day task in drug development, this tool may be useful in a number of projects. It requires two files with molecules, a fingerprint identifier, a similarity measure, and the processing mode as input. Fingerprint identifiers describe which fingerprint should be generated and, in the case of ECFP and FCFP, also the maximal bond diameter (see section 5.1.1), e.g., ECFP_2, ECFP_4, ECFP_6, etc. The similarity measure is either one of Tanimoto, Cosine, Hamming, Euclidian, or Dice (see section 5.2.1). Molecules are compared in a pairwise fashion. A molecule from the first file is compared to all molecules from the second file. The processing mode determines which value is written to the output:

all All similarity values with a similarity greater than a user specified cutoff are generated. The cutoff value is specified via the *–cutoff* parameter.

best Only the highest similarity value for a given molecule of the first set and all molecules of the second set are output. The *–cutoff* parameter has no effect in this case.

If no file is given as output parameter, similarity values are written to *cout*. The fingerprint tool uses the Intel Threading Building Blocks library[21] for parallelization. An optional parameter is therefore the number of threads; a single thread is used per default. A complete list of command line parameters can be found in Appendix A.5.

9.6 Other Tools

This section lists a few command line tools that were developed during this project and became part of the NAOMI repository. The fragment space tools were developed for convenience and are not directly relevant to the experiments presented here. The PMI (principle moment of inertia) tool was developed while evaluating different methods for the characterization of large sets of molecules, although, it was not used in the end.

[21] https://www.threadingbuildingblocks.org

fragspace_converter converts a FSF formatted fragment space file into FSDB.

fragspace_creator creates a fragment space file from a list of fragments, connection rules, and terminators.

fragspace_info prints information about a given fragment space in FSF or FSDB format.

molecule_calc_pmi calculates normalized PMI ratios (NPR) for a list of molecules. The principle moment of inertia (PMI) can be used to characterize the shape of a molecule [137]. It either uses the 3D coordinates from a file or generates coordinates. In the latter case, two methods are offered: A simple NAOMI procedure that generates a clash free initial conformation and a more elaborate method for generating conformational ensembles [138].

10 Conclusion and Outlook

In this thesis, two new algorithms and a graphical user interface were developed based on fragment spaces as a representation of combinatorial chemical space. One of the algorithms was implemented as a proof-of-concept and represents a completely novel method for generating fragment spaces in a fully-automated fashion by applying synthetic chemical reactions to a set of input molecules. The other algorithm represents a deterministic and exhaustive enumeration strategy for molecules with a user-defined physicochemical profile, implemented with constant main memory consumption to allow enumeration of large chemical spaces. The graphical user interface, Fragment Space Commander, enables the user to conveniently work with fragment spaces by creating, modifying, and searching them. All three parts were implemented based on the fragment space implementation of the NAOMI cheminformatics library [24, 59]. It was shown how these tools can be of use in rational drug design and aid the development of new molecules with desirable physicochemical properties. In the following, each tool is summarized and improvements are discussed.

In addition to the three main research subjects, several smaller components and tools were developed in order to facilitate the work done in this project. Consequently, they became part of the NAOMI source code repository. This includes the implementation of an existing algorithm for retrosynthetic fragmentation of molecules [34] that was used for data generation, several fingerprint methods [111, 119] and similarity measures [117] that were used to analyze enumerated libraries, as well as additional helper tools.

10.1 Enumerator

The goal of this project was to develop an algorithm for the enumeration of molecules with user-defined physicochemical properties from fragment spaces. This is challenging due to handling of redundancy and the fact that such a chemical space can describe a virtually infinite number of molecules. The FSees algorithm was newly implemented utilizing the molecule model of NAOMI [59] and designed to overcome the limitations of a previous prototype algorithm based on the Flex* library [29]. This ability to enumerate large chemical space is accomplished by using file-based data structures instead of main memory. Furthermore, efficient strategies to avoid and detect redundant molecules were developed. For the former, symmetry in fragments was considered for the first time, for the latter, an efficient database with a Bloom-filter was used to keep track of duplicates.

In three experiments, a total of 105 fragment spaces were generated by fragmentation [34] and enumerated with different sets of constraints (section 8.2). The first experiment was a large scale enumeration of fragment spaces generated from 102 inhibitor classes (section 8.2.3). The enumerated molecules and a general purpose screening library were compared to known active molecules regarding their similarity. It could be observed that the enumerated molecules contain a higher ratio of closely related molecules than the general purpose library and are therefore a source of promising lead structures. In the second experiment, an alternative enumeration mode was applied in two cases (section 8.2.4). Both were modeled after a typical fragment-based design problem and demonstrate the usefulness of the algorithm in this context. For the last experiment, a very large fragment space was built from approved drug-molecules first (section 8.2.5). Then, this space was enumerated with lead-like properties generating a library of 0.5 billion molecules named Hamburg enumerated lead-like set [1]. This library contains mostly novel compounds and therefore represents an interesting source for lead discovery projects.

Validation of enumerated molecules was done purely based on molecular similarity. In order to asses the quality of the results better, the next logical step – with respect to the (virtual) drug discovery pipeline – would be to subject them to virtual screening. This was unfortunately not within the scope of this thesis (see introduction of chapter 5).

As demonstrated by the experiments, the algorithm works well in the discussed scenarios and is able to enumerate very large spaces. The largest single enumeration, i.e., DUD-E VGFR2, yielded 113,625,393 molecules and took 82.3 hours. Yet, there are aspects of the algorithm that could be improved. In the following, it is distinguished between conceptual and technical improvements. The former, which describes changes to the logic of the algorithm, are discussed first. A first improvement is the handling of symmetry in more complex cases, i.e., fragments with more than three linkers and the detection of symmetry in intermediate products (see Figure 6.11). Furthermore, the repertoire of enumeration strategies as described in section 6.4.8 could be extended: For instance, a fragment could be limited to occur in a molecule only up to a defined number of times or a fragment cannot be connected to itself (see ENUM299192 in Figure 8.13 c). Technical improvements concern the implementation and are equivalent to performance improvements. Currently, one of the bottlenecks is the processing power of a CPU core because a single thread is used for all computations. A sensible strategy to increase the rate of molecule generation is to use threading. In this case, it must be ensured that the database is always current because it is used for redundancy checking. The second bottleneck is database performance. Because the only task of the database is to efficiently determine whether an object is present or not, a relational model as provided by SQLite is not required and provides a certain amount of computational overhead. A simple key-value store or some other type of NoSQL database may be better suited for this use case.

10.2 Reaction-based Fragment Spaces

The generation of fragment spaces from reactions and synthetically available molecules represents an attractive approach for rational drug design as it eventually results in molecules that are likely to be synthesizable in the laboratory. Previous methods were either based on retrosynthetic fragmentation [34] or used reaction information indirectly by combining combinatorial chemistry libraries (CoLibri [106]). The goal of this project was to automize the matching and annotation process in order to conveniently and quickly generate custom fragment spaces from building block libraries. By constructing a fragment space, the reaction information need only to be used once rather than every time the tool is executed, like in other methods [20, 21, 99]. Furthermore, fragment spaces can be used by a variety of methods [24, 25, 27, 29, 31, 32]. In a validation experiment, it was shown how fragment spaces representing combinatorial libraries can be constructed and how the same input can be used to construct a combined fragment space representing 22 different reactions.

The algorithm was implemented as a proof-of-concept and is limited to relatively simple organic reactions. Naturally, there is much room to improve this implementation. First, the reaction matching based on SMIRKS should be made more robust to overcome the limitations discussed in section 8.3, i.e., bond changes are only considered at the connecting atom. Second, custom terminators should be used that are built from atoms or concrete functional groups that are split off during a reaction. Since this is not considered at the time, default terminators are used, hence a fragment may be terminated with a different fragment as the initial one. This may result in molecules with missing groups. Third, a strategy for the automatic assignment of link types should be added. Currently, link types are a required user-defined input that is read from the same file as reactions. Fourth, the tool could be parallelized using threading. This could be realized easily by incorporating the Intel Threading Building Blocks library[22] – as used in other tools (see chapter 9) – because molecules can be processed independently.

The biggest challenge is the inclusion of ring forming reactions. 50% of the reactions published by Hartenfeller et al. are ring forming reactions [96] and can currently not be considered. During a simple reaction, only one bond is formed when two fragments are connected. In the case of ring formation, at least two new bonds are usually formed (see Figure 10.1 a and b). In order to model this, pairs of linkers must be dependent in the fragment space compatibility description. This would require a significant change to the fragment space data structure to accommodate complex compatibility definitions and the adjustment of existing algorithms, which usually utilize a tree data structure to model molecules [24, 25, 27, 29, 31, 32]. However, ring forming reactions can be modeled differently. Instead of connecting fragments via two linker pairs, the ring resulting from the reaction is stored as a *ring fragment*. Linkers are added to the ring fragment at the position where it connects to the rest of the initial fragment, i.e., the part that is not involved in the reaction. In other words, all atoms that are involved in the reaction are cleaved off and the fragment is annotated with a linker only compatible to the ring fragment. An example is shown in Figure 10.1: Instead of adding multiple linkers to the reactive atoms in the reactants (see left side of the reaction equation in a and b), an additional fragment is introduced (c, middle) that serves as a linking fragment. This strategy would require no changes to existing algorithms, nor the fragment space data structure, but requires some more advanced construction algorithm.

10.3 Fragment Space Commander

Graphical user interfaces represent a convenient way to access scientific software and provide interested parties without computational background with an easy to understand interface. Fragment Space Commander (FSC) is a graphical user interface to visualize, create, modify, and query fragment spaces utilizing various methods, which was developed within the scope of this thesis. It provides access to the fragment space framework that is part of the NAOMI library and enables the user to interactively work with fragment spaces in a single program for the first time. Fragment spaces can be created by shredding of molecules or by applying

[22] https://www.threadingbuildingblocks.org

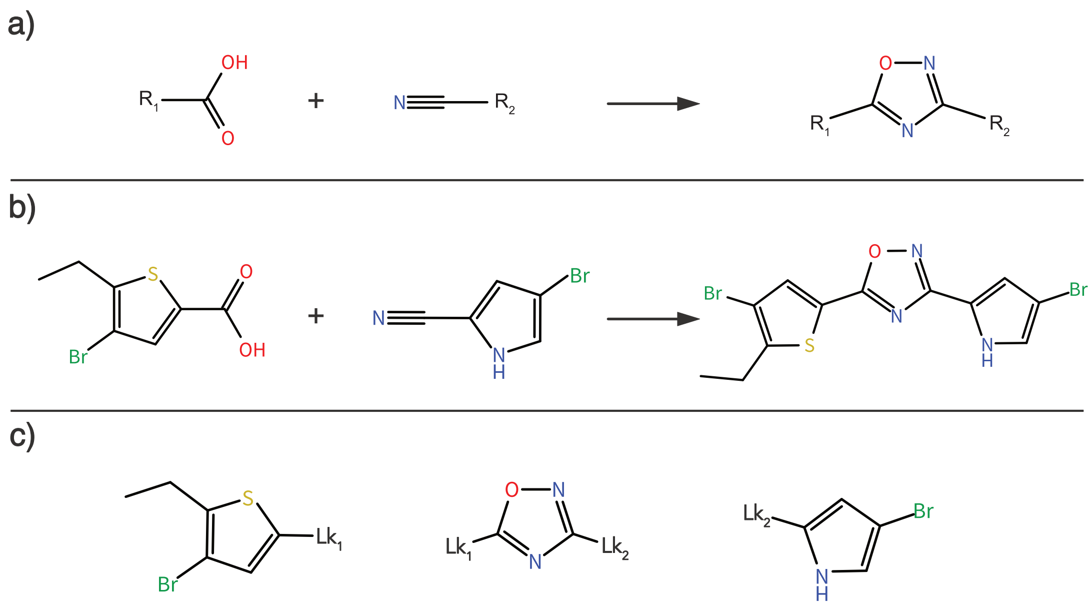


Figure 10.1: Example for a ring forming reaction, i.e., synthesis of oxadiazole according to Hartenfeller et al. [96]. a) Depiction of the atoms involved in the reaction except that another nitrogen is added, which is not accounted for on the reactant side. The reaction has been simplified for computational processing, as discussed in 6.2.1. b) Example reaction with molecules from the ChemBridge Building Blocks (see section 7.1.6). c) Fragments necessary to model this reaction in a fragment space. The fragment in the middle is introduced as a linking fragment, connecting the two reactive molecules.

synthetic reactions to molecules. After creating a fragment space or loading it from a file, its content (fragments and connection rules) is visualized and can be edited by applying various filters. Aside from providing access to the constraint-based enumeration algorithm, query-based search methods – similarity [24] and substructure searching [32, 33] – can be accessed as well. Once result molecules are available, they can be browsed directly in FSC and saved to a file.

In order to improve FSC, the data entry process could be enhanced. Currently, cut rules and reactions must be pasted into several fields as SMARTS and SMIRKS, respectively. A wizard based system that visualizes – and possibly allows to edit – cut rules and reactions would be much more convenient. Another improvement could be the addition of a task manager and progress bar showing how far along the respective processes are. In this context, it would be interesting to implement a client-server design as used by Molpher [139]. Since several computations take more than a few seconds to compute, it might be desirable to transfer the heavy processing to a dedicated computer with more processors and memory. FSC could also be integrated with existing tools from the NAOMI framework, such as MONA [63, 66] or SMARTSEditor [140]. For example, it would be possible to store generated molecules directly in a MONA database file instead of SMILES, Mol2 or SDF files from where they have to be read and processed again.

10.3.1 Storing Fragment Spaces

The FSDB format was designed in the FSC context to replace the outdated and error-prone FSF format and allow for interactive filtering of fragments. It does have several advantages as discussed in section 9.1.5, but towards the end of the project, it became apparent that it is not ideal for exchanging information. It provides the basis for Fragment Space Commander and as such could be developed further, for example, to store multiple fragment spaces so that only one database instance is required. However, in order to modify or even merely view the raw data in an FSDB file, a SQLite client and basic knowledge about SQL is required, which may not be ideal for many users. To reliably store the information of a fragment space, a human-readable, text-based format such as XML is much more convenient. This would allow to make simple modifications to fragment spaces if manual adjustments were necessary.

Bibliography

[1] Florian Lauck and Matthias Rarey. “FSees: Customized Enumeration of Chemical Subspaces with Limited Main Memory Consumption”. In: *Journal of Chemical Information and Modeling* 56.9 (Sept. 2016), pp. 1641–1653.

[2] Florian Lauck and Matthias Rarey. “Coping with Combinatorial Space in Molecular Design”. In: *De novo Molecular Design*. Weinheim, Germany: Wiley-VCH Verlag GmbH & Co. KGaA, Oct. 2013, pp. 325–347.

[3] Florian Lauck. “Exploring local chemical space: From fragments to molecules with a defined physicochemical profile”. In: *Fragment-based Lead Discovery Conference*. Basel, Switzerland, Sept. 2014.

[4] Florian Lauck and Matthias Rarey. “Dealing with Combinatorial Chemical Space: Towards a Universal Framework”. In: *6th Joint Sheffield Conference on Chemoinformatics*. Sheffield, United Kingdom, July 2013.

[5] Florian Lauck and Matthias Rarey. “Design Your Chemical Universe: Mining Fragment Spaces for Novel Molecules”. In: *20th European Symposium on Quantitative Structure-Activity Relationship*. St. Petersburg, Russia, Sept. 2014.

[6] Gerhard Klebe. *Wirkstoffdesign*. Entwurf und Wirkung von Arzneistoffen. Heidelberg: Spektrum Akademischer Verlag, 2009.

[7] Gisbert Schneider and Uli Fechner. “Computer-Based De Novo Design of Drug-Like Molecules.” In: *Nature Reviews Drug discovery* 4.8 (Aug. 2005), pp. 649–663.

[8] Peter S Kutchukian and Eugene I Shakhnovich. “De novo design: balancing novelty and confined chemical space”. In: *Expert Opinion on Drug Discovery* 5.8 (Aug. 2010), pp. 789–812.

[9] Regine S Bohacek, Colin McMartin, and Wayne C Guida. “The Art and Practice of Structure-Based Drug Design: A Molecular Modeling Perspective”. In: *Medicinal Research Reviews* 16.1 (1996), pp. 3–50.

[10] W P Walters, M T Stahl, and M A Murcko. “Virtual screening-an overview”. In: *Drug Discovery Today* 3 (1998), pp. 160–178.

[11] Kurt L M Drew et al. “Size estimation of chemical space: how big is it?” In: *Journal of Pharmacy and Pharmacology* 64.4 (Apr. 2012), pp. 490–495.

[12] Simon J Teague et al. “The Design of Leadlike Combinatorial Libraries”. In: *Angewandte Chemie International Edition* 38.24 (Dec. 1999), pp. 3743–3748.

[13] Arup K Ghose, Vellarkad N Viswanadhan, and John J Wendoloski. “A Knowledge-Based Approach in Designing Combinatorial or Medicinal Chemistry Libraries for Drug Discovery. 1. A Qualitative and Quantitative Characterization of Known Drug Databases”. In: *Journal of Chemical Documentation* 1.1 (Jan. 1999), pp. 55–68.

[14] T I Oprea et al. “Is There a Difference between Leads and Drugs? A Historical Perspective”. In: *Journal of Chemical Information and Modeling* 41.5 (Sept. 2001), pp. 1308–1315.

[15] Christopher A Lipinski et al. “Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings”. In: *Advanced Drug Delivery Reviews* 46.1–3 (Jan. 2001), pp. 3–26.

[16] Daniel F Veber et al. “Molecular Properties That Influence the Oral Bioavailability of Drug Candidates”. In: 45.12 (May 2002), pp. 2615–2623.

[17] Andreas Reichel. “The Role of Blood-Brain Barrier Studies in the Pharmaceutical Industry.” In: *Current drug metabolism* 7.2 (Feb. 2006), pp. 183–203.

[18] Oleg Ursu et al. “Understanding drug-likeness”. In: *Wiley Interdisciplinary Reviews: Computational Molecular Science* 1.5 (Apr. 2011), pp. 760–781.

[19] G Richard Bickerton et al. “Quantifying the Chemical Beauty of Drugs.” In: *Nature Chemistry* 4.2 (Feb. 2012), pp. 90–98.

[20] H Maarten Vinkers et al. “SYNOPSIS: SYNthesize and OPtimize System in Silico.” In: *Journal of Medicinal Chemistry* 46.13 (June 2003), pp. 2765–2773.

[21] Markus Hartenfeller et al. “DOGS: Reaction-Driven de novo Design of Bioactive Compounds”. In: *Plos Computational Biology* 8.2 (2012), e1002380.

[22] Kathryn Loving, Ian Alberts, and Woody Sherman. “Computational Approaches for Fragment-Based and De Novo Design.” In: *Current Topics in Medicinal Chemistry* 10.1 (2010), pp. 14–32.

[23] Diane Joseph-McCarthy et al. “Fragment-Based Lead Discovery and Design”. In: *Journal of Chemical Information and Modeling* 54.3 (Mar. 2014), pp. 693–704.

[24] Matthias Rarey and Martin Stahl. "Similarity searching in large combinatorial chemistry spaces." In: *Journal of computer-aided molecular design* 15.6 (June 2001), pp. 497–520.

[25] Jörg Degen and Matthias Rarey. "FlexNovo: structure-based searching in large fragment spaces. - PubMed - NCBI". In: *ChemMedChem* 1.8 (Aug. 2006), pp. 854–868.

[26] Jörg Degen. "Entwicklung eines strukturbasierten de-novo-Design-Verfahrens und chemischer Fragmenträume für den rationalen Wirkstoffentwurf". PhD thesis. Apr. 2009.

[27] Patrick Maaß et al. "Recore: A Fast and Versatile Method for Scaffold Hopping Based on Small Molecule Crystal Structure Conformations". In: *Journal of Chemical Information and Modeling* 47.2 (2007), pp. 390–399.

[28] Patrick Maaß. "Neue rechnergestützte Methoden zum Scaffold-Hopping unter Verwendung von Kristallstrukturen kleiner Moleküle". PhD thesis. Feb. 2009.

[29] Juri Pärn, Jörg Degen, and Matthias Rarey. "Exploring fragment spaces under multiple physicochemical constraints". In: *Journal of computer-aided molecular design* 21.6 (2007), pp. 327–340.

[30] Juri Pärn. "Eigenschaftsbasierte Auswahl von Molekülen aus chemischen Fragmenträumen". PhD thesis. Universität Hamburg, Oct. 2003.

[31] Hans-Christian Ehrlich, Andrea Volkamer, and Matthias Rarey. "Searching for Substructures in Fragment Spaces". In: *Journal of Chemical Information and Modeling* 52.12 (2012), pp. 3181–3189.

[32] Hans-Christian Ehrlich, Angela M Henzler, and Matthias Rarey. "Searching for Recursively Defined Generic Chemical Patterns in Nonenumerated Fragment Spaces". In: *Journal of Chemical Information and Modeling* 53.7 (2013), pp. 1676–1688.

[33] Hans-Christian Ehrlich. "Searching for Generic Chemical Patterns in Combinatorial Chemical Spaces". PhD thesis. Universität Hamburg, Oct. 2013.

[34] Jörg Degen et al. "On the Art of Compiling and Using 'Drug-Like' Chemical Fragment Spaces." In: *ChemMedChem* 3.10 (Oct. 2008), pp. 1503–1507.

[35] Carsten Detering et al. "KnowledgeSpace - a publicly available virtual chemistry space". In: *Journal of cheminformatics.* 2010, O9.

[36] Kevin E Knockenhauer and Thomas U Schwartz. "The Nuclear Pore Complex as a Flexible and Dynamic Gate". In: *Cell* 164.6 (Mar. 2016), pp. 1162–1171.

[37] Rolf Knippers. “Molekulare Genetik”. In: (2001).

[38] Albert L Lehninger, David L Nelson, and Michael M Cox. *Prinzipien der Biochemie*. Ed. by Harald Tschesche. Spektrum Lehrbuch. Heidelberg [u.a.]: Spektrum Akad. Verl, 1998.

[39] Peter Imming, Christian Sinning, and Achim Meyer. “Drugs, their targets and the nature and number of drug targets”. In: *Nature Reviews Drug discovery* 5.10 (Oct. 2006), pp. 821–834.

[40] Mathias Rask-Andersen, Markus Sällman Almén, and Helgi B Schiöth. “Trends in the exploitation of novel drug targets”. In: *Nature Reviews Drug discovery* 10.8 (Aug. 2011), pp. 579–590.

[41] H M Berman et al. “The Protein Data Bank.” In: *Nucleic acids research* 28.1 (Jan. 2000), pp. 235–242.

[42] Christopher Lipinski and Andrew Hopkins. “Navigating chemical space for biology and medicine”. In: *Nature* 432.7019 (Dec. 2004), pp. 855–861.

[43] Steven M Paul et al. “How to improve R&D productivity: the pharmaceutical industry’s grand challenge”. In: *Nature Reviews Drug discovery* 9.3 (Mar. 2010), pp. 203–214.

[44] David C Rees et al. “Fragment-Based Lead Discovery”. In: *Nature Reviews Drug discovery* 3.8 (Aug. 2004), pp. 660–672.

[45] Daniel A Erlanson, Robert S McDowell, and Tom O’Brien. “Fragment-Based Drug Discovery”. In: *Journal of Medicinal Chemistry* 47.14 (June 2004), pp. 3463–3482.

[46] Robin A E Carr et al. “Fragment-based lead discovery: leads by design”. In: *Drug Discovery Today* 10.14 (July 2005), pp. 987–992.

[47] Philip J Hajduk and Jonathan Greer. “A decade of fragment-based drug design: strategic advances and lessons learned.” In: *Nature Reviews Drug discovery* 6.3 (Mar. 2007), pp. 211–219.

[48] Christopher W Murray and David C Rees. “The rise of fragment-based drug discovery”. In: *Nature Chemistry* 1.3 (June 2009), pp. 187–192.

[49] Monya Baker. “Fragment-based lead discovery grows up”. In: *Nature Reviews Drug discovery* 12.1 (Jan. 2013), pp. 5–7.

[50] Miles Congreve et al. “A ’rule of three’ for fragment-based lead discovery?” In: *Drug Discovery Today* 8.19 (Oct. 2003), pp. 876–877.

[51] Christopher M Dobson. “Chemical space and biology”. In: *Nature* 432.7019 (Dec. 2004), pp. 824–828.

[52] URL. *Chemical Substances - CAS REGISTRY*. URL: http://www.cas.org/content/chemical-substances.

[53] Melvin J Yu. “Natural product-like virtual libraries: recursive atom-based enumeration.” In: *Journal of Chemical Documentation* 51.3 (Mar. 2011), pp. 541–557.

[54] Jean-Louis Reymond et al. “The enumeration of chemical space”. In: *Wiley Interdisciplinary Reviews: Computational Molecular Science* 2.5 (2012), pp. 717–733.

[55] Zhengwei Peng. “Very large virtual compound spaces: construction, storage and utility in drug discovery”. In: *Drug Discovery Today: Technologies* 10.3 (Sept. 2013), e387–e394.

[56] Gisbert Schneider et al. “De novo design of molecular architectures by evolutionary assembly of drug-derived building blocks”. In: *Journal of computer-aided molecular design* 14.5 (2000), pp. 487–494.

[57] Xiao Qing Lewell et al. “RECAP-Retrosynthetic Combinatorial Analysis Procedure: A Powerful New Technique for Identifying Privileged Molecular Fragments with Useful Applications in Combinatorial Chemistry”. In: *Journal of Chemical Information and Computer Sciences* 38.3 (1998), pp. 511–522.

[58] Matthias Rarey and J S Dixon. “Feature trees: A new molecular similarity measure based on tree matching.” In: *Journal of computer-aided molecular design* 12.5 (Sept. 1998), pp. 471–490.

[59] Sascha Urbaczek et al. “NAOMI: on the almost trivial task of reading molecules from different file formats.” In: *Journal of Chemical Information and Modeling* 51.12 (Dec. 2011), pp. 3199–3207.

[60] Sascha Urbaczek, Adrian Kolodzik, and Matthias Rarey. “The Valence State Combination Model: A Generic Framework for Handling Tautomers and Protonation States”. In: *Journal of Chemical Information and Modeling* 54.3 (Mar. 2014), pp. 756–766.

[61] Stefan Bietz et al. “Protoss: a holistic approach to predict tautomers and protonation states in protein-ligand complexes”. In: *Journal of cheminformatics* 6.1 (Apr. 2014), p. 1.

[62] Karen Schomburg et al. “From Structure Diagrams to Visual Chemical Patterns”. In: *Journal of Chemical Documentation* 50.9 (Sept. 2010), pp. 1529–1535.

[63] Matthias Hilbig et al. “MONA - Interactive manipulation of molecule collections.” In: *Journal of cheminformatics* 5.1 (2013), pp. 1–10.

[64] Karen T Schomburg and Matthias Rarey. “Benchmark Data Sets for Structure-Based Computational Target Prediction”. In: *Journal of Chemical Information and Modeling* 54.8 (Aug. 2014), pp. 2261–2274.

[65] Angela M Henzler et al. “An integrated approach to knowledge-driven structure-based virtual screening.” In: *Journal of computer-aided molecular design* 28.9 (Sept. 2014), pp. 927–939.

[66] Matthias Hilbig and Matthias Rarey. “MONA 2: A Light Cheminformatics Platform for Interactive Compound Library Processing.” In: *Journal of Chemical Documentation* 55.10 (Oct. 2015), pp. 2071–2078.

[67] Stefan Bietz and Matthias Rarey. “ASCONA: Rapid Detection and Alignment of Protein Binding Site Conformations.” In: *Journal of Chemical Documentation* 55.8 (Aug. 2015), pp. 1747–1756.

[68] Stefan Bietz et al. “Discriminative Chemical Patterns: Automatic and Interactive Design”. In: *Journal of Chemical Information and Modeling* 55.8 (Aug. 2015), pp. 1535–1546.

[69] Stefan Bietz and Matthias Rarey. “SIENA: Efficient Compilation of Selective Protein Binding Site Ensembles.” In: *Journal of Chemical Documentation* 56.1 (Jan. 2016), pp. 248–259.

[70] Dominique Douguet, Etienne Thoreau, and Gérard Grassy. “A genetic algorithm for the automated generation of small organic molecules: Drug design using an evolutionary algorithm”. In: *Journal of computer-aided molecular design* 14.5 (2000), pp. 449–466.

[71] Scott C-H Pegg, Jose J Haresco, and Irwin D Kuntz. “A genetic algorithm for structure-based de novo design”. In: *Journal of computer-aided molecular design* 15.10 (2001), pp. 911–933.

[72] Eric-Wubbo Lameijer et al. “The molecule evoluator. An interactive evolutionary algorithm for the design of drug-like molecules.” In: *Journal of Chemical Information and Modeling* 46.2 (Mar. 2006), pp. 545–552.

[73] Aaron M Virshup et al. “Stochastic Voyages into Uncharted Chemical Space Produce a Representative Library of All Possible Drug-Like Compounds”. In: *Journal of the American Chemical Society* 135.19 (May 2013), pp. 7296–7303.

[74] Kentaro Kawai, Naoya Nagata, and Yoshimasa Takahashi. “De Novo Design of Drug-Like Molecules by a Fragment-Based Molecular Evolutionary Approach”. In: *Journal of Chemical Information and Modeling* 54.1 (Jan. 2014), pp. 49–56.

[75] Chetan Rupakheti et al. “Strategy To Discover Diverse Optimal Molecules in the Small Molecule Universe”. In: *Journal of Chemical Information and Modeling* 55.3 (Feb. 2015), pp. 529–537.

[76] Xiangqian Hu, David N Beratan, and Weitao Yang. “A gradient-directed Monte Carlo approach to molecular design.” In: *The Journal of chemical physics* 129.6 (Aug. 2008), p. 064102.

[77] Raymond E Carhart et al. “Applications of artificial intelligence for chemical inference. 37. GENOA: a computer program for structure elucidation utilizing overlapping and alternative substructures”. In: *The Journal of Organic Chemistry* 46.8 (May 2002), pp. 1708–1718.

[78] Kimito Funatsu, Nobuyoshi Miyabayashi, and Shinichi Sasaki. “Further Development of Structure Generation in the Automated Structure Elucidation System CHEMICS”. In: *Journal of Chemical Information and Modeling* 28.1 (Feb. 1988), pp. 18–28.

[79] M S Molchanova, V V Shcherbukhin, and N S Zefirov. “Computer Generation of Molecular Structures by the SMOG Program”. In: *Journal of Chemical Information and Modeling* 36.4 (July 1996), pp. 888–899.

[80] Martin Badertscher et al. “Assemble 2.0: A Structure Generator”. In: *Drug Discovery Today: Technologies* 51.1 (May 2000), pp. 73–79.

[81] J L Faulon, C J Churchwell, and D P Visco. “The signature molecular descriptor. 2. Enumerating molecules from their extended valence sequences”. In: *Journal of Chemical Information and Computer Sciences* 43.3 (2003), pp. 721–734.

[82] A Korytko et al. “HOUDINI: A New Approach to Computer-Based Structure Generation”. In: *Journal of Chemical Information and Modeling* 43.5 (Sept. 2003), pp. 1434–1446.

[83] Julio E Peironcely et al. “OMG: open molecule generator”. In: *Journal of cheminformatics* 4.1 (2012), p. 21.

[84] David S Wishart et al. “DrugBank: a comprehensive resource for in silico drug discovery and exploration.” In: *Nucleic Acids Res.* 34.Database issue (Jan. 2006), pp. D668–72.

[85] Tobias Fink, Heinz Bruggesser, and Jean-Louis Reymond. “Virtual Exploration of the Small-Molecule Chemical Universe below 160 Daltons”. In: *Angewandte Chemie International Edition* 44.10 (2005), pp. 1504–1508.

[86] Lorenz C Blum and Jean-Louis Reymond. “970 Million Druglike Small Molecules for Virtual Screening in the Chemical Universe Database GDB-13.” In: *Journal of the American Chemical Society* 131.25 (July 2009), pp. 8732–8733.

[87] Lars Ruddigkeit et al. “Enumeration of 166 Billion Organic Small Molecules in the Chemical Universe Database GDB-17”. In: *Journal of Chemical Documentation* 52.11 (Nov. 2012), pp. 2864–2875.

[88] Jean-Louis Reymond and Mahendra Awale. “Exploring Chemical Space for Drug Discovery Using the Chemical Universe Database”. In: *ACS Chemical Neuroscience* 3.9 (Jan. 2012), pp. 649–657.

[89] Lorenz C Blum et al. “Discovery of α7-nicotinic Receptor Ligands by Virtual Screening of the Chemical Universe Database GDB-13.” In: *Journal of Chemical Documentation* 51.12 (Dec. 2011), pp. 3105–3112.

[90] Lorenz C Blum, Ruud van Deursen, and Jean-Louis Reymond. “Visualisation and Subsets of the Chemical Universe Database GDB-13 for Virtual Screening.” In: *Journal of computer-aided molecular design* 25.7 (July 2011), pp. 637–647.

[91] Erika Luethi et al. “Identification of Selective Norbornane-Type Aspartate Analogue Inhibitors of the Glutamate Transporter 1 (Glt-1) from the Chemical Universe Generated Database (GDB).” In: *Journal of Medicinal Chemistry* 53.19 (Oct. 2010), pp. 7236–7250.

[92] N Garcia-Delgado et al. “Exploring a7-nicotinic Receptor Ligand Diversity by Scaffold Enumeration from the Chemical Universe Database GDB”. In: *ACS Medicinal Chemistry Letters* 1 (2010), pp. 422–426.

[93] Kong Thong Nguyen et al. “3-(aminomethyl)piperazine-2,5-dione as a novel NMDA glycine site inhibitor from the chemical universe database GDB.” In: *Bioorganic & medicinal chemistry letters* 19.14 (July 2009), pp. 3832–3835.

[94] Kong Thong Nguyen et al. “Discovery of NMDA Glycine Site Inhibitors from the Chemical Universe Database GDB”. In: *ChemMedChem* 3.10 (2008), pp. 1520–1524.

[95] Brendan D McKay. “Isomorph-Free Exhaustive Generation”. In: *Journal of Algorithms* 26.2 (Feb. 1998), pp. 306–324.

[96] Markus Hartenfeller et al. “A collection of robust organic synthesis reactions for in silico molecule design.” In: *Journal of Chemical Documentation* 51.12 (Dec. 2011), pp. 3093–3098.

[97] Valerie J Gillet et al. “Designing focused libraries using MoSELECT”. In: *Journal of Molecular Graphics & Modelling* 20.6 (June 2002), pp. 491–498.

[98] Jacob D Durrant and J Andrew McCammon. “AutoClickChem: Click Chemistry in Silico”. In: *Plos Computational Biology* 8.3 (2012), e1002397.

[99] Stephan C Schürer, Prashant Tyagi, and Steven M Muskal. “Prospective exploration of synthetically feasible, medicinally relevant chemical space.” In: *Journal of Chemical Information and Modeling* 45.2 (Mar. 2005), pp. 239–248.

[100] *Daylight Theory Manual.* 4.9. Laguna Niguel, CA, Aug. 2011.

[101] F Chevillard and P Kolb. “SCUBIDOO: A Large yet Screenable and Easily Searchable Database of Computationally Created Chemical Compounds Optimized toward High Likelihood of Synthetic Tractability”. In: *Journal of Chemical Information and Modeling* 55.9 (Sept. 2015), pp. 1824–1835.

[102] *ChemBridge | Building Blocks.* URL: http://www.chembridge.com/building_blocks/.

[103] Uli Fechner and Gisbert Schneider. “Flux (1): A Virtual Synthesis Scheme for Fragment-Based de Novo Design”. In: *J. Chem. Inf. Comput. Sci.* 46.2 (Dec. 2005), pp. 699–707.

[104] Uli Fechner and Gisbert Schneider. “Flux (2): Comparison of Molecular Mutation and Crossover Operators for Ligand-Based de Novo Design”. In: *Journal of Chemical Information and Modeling* 47.2 (Feb. 2007), pp. 656–667.

[105] Markus Hartenfeller et al. “Concept of Combinatorial De NovoDesign of Drug-like Molecules by Particle Swarm Optimization”. In: *Chemical Biology & Drug Design* 72.1 (July 2008), pp. 16–26.

[106] Markus Boehm et al. “Similarity Searching and Scaffold Hopping in Synthetically Accessible Combinatorial Chemistry Spaces”. In: *Journal of Medicinal Chemistry* 51.8 (2008), pp. 2468–2480.

[107] Uta Lessel et al. “Searching Fragment Spaces with Feature Trees”. In: *Journal of Chemical Information and Modeling* 49.2 (Feb. 2009), pp. 270–279.

[108] Sergey Nikitin et al. “A very large diversity space of synthetically accessible compounds for use with drug design programs”. In: *Journal of computer-aided molecular design* 19.1 (2005), pp. 47–63.

[109] J Robert Fischer, Uta Lessel, and Matthias Rarey. “LoFT: Similarity-Driven Multiobjective Focused Library Design”. In: *Journal of Chemical Information and Modeling* 50.1 (2010), pp. 1–21.

[110] Roberto Todeschini and Viviana Consonni. *Handbook of Molecular Descriptors.* Ed. by Raimund Mannhold, Henk Timmerman, and Hugo Kubinyi. Methods and Principles in Medicinal Chemistry. Weinheim, Germany: Wiley-VCH Verlag GmbH, Sept. 2000.

[111] David Rogers and Mathew Hahn. “Extended-connectivity fingerprints.” In: *Journal of Chemical Information and Modeling* 50.5 (May 2010), pp. 742–754.

[112] Raimund Mannhold et al. “Calculation of molecular lipophilicity: State-of-the-art and comparison of log P methods on more than 96,000 compounds”. In: *Journal of Pharmaceutical Sciences* 98.3 (Mar. 2009), pp. 861–893.

[113] David Weininger. “SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules”. In: *Journal of Chemical Information and Modeling* 28.1 (Feb. 1988), pp. 31–36.

[114] David Weininger, Arthur Weininger, and Joseph L Weininger. “SMILES. 2. Algorithm for generation of unique SMILES notation”. In: *Journal of Chemical Information and Modeling* 29.2 (May 1989), pp. 97–101.

[115] Thomas H Cormen et al. “Introduction to algorithms”. In: (2009).

[116] Hans-Christian Ehrlich and Matthias Rarey. “Maximum common subgraph isomorphism algorithms and their applications in molecular science: a review”. In: *Wiley Interdisciplinary Reviews: Computational Molecular Science* 1.1 (Jan. 2011), pp. 68–79.

[117] P Willett, J M Barnard, and Geoffrey M Downs. “Chemical Similarity Searching”. In: *Journal of Chemical Information and Computer Sciences* 38.6 (Nov. 1998), pp. 983–996.

[118] Gerald M Maggiora and Veerabahu Shanmugasundaram. “Molecular similarity measures.” In: *Methods in molecular biology (Clifton, N.J.)* 672 (2011), pp. 39–100.

[119] R Nilakantan et al. “Topological torsion: a new molecular descriptor for SAR applications. Comparison with other descriptors”. In: *Journal of Chemical Information and Computer Sciences* 27.2 (1987), pp. 82–85.

[120] J Kazius, R McGuire, and R Bursi. “Derivation and validation of toxicophores for mutagenicity prediction”. In: *Journal of Medicinal Chemistry* 48.1 (2005), pp. 312–320.

[121] Hans-Christian Ehrlich and Matthias Rarey. “Systematic benchmark of substructure search in molecular graphs - From Ullmann to VF2”. In: *Journal of cheminformatics* 4.1 (2012), p. 13.

[122] S A Wildman and G M Crippen. “Prediction of Physicochemical Parameters by Atomic Contributions”. In: *Journal of Chemical Information and Modeling* 39.5 (Sept. 1999), pp. 868–873.

[123] Richard Hipp. *SQLite.* URL: https://www.sqlite.org/.

[124] Stephen Heller et al. “InChI - the worldwide chemical structure identifier standard”. In: *Journal of cheminformatics* 5.1 (2013), pp. 1–9.

[125] Burton H Bloom. “Space/time trade-offs in hash coding with allowable errors”. In: *Communications of the ACM* 13.7 (1970), pp. 422–426.

[126] Anna Gaulton et al. “ChEMBL: a large-scale bioactivity database for drug discovery.” In: *Nucleic Acids Res.* 40.Database issue (Jan. 2012), pp. D1100–7.

[127] A Patricia Bento et al. “The ChEMBL bioactivity database: an update.” In: *Nucleic Acids Res.* 42.Database issue (Jan. 2014), pp. D1083–90.

[128] George Papadatos et al. “SureChEMBL: a large-scale, chemically annotated patent document database.” In: *Nucleic Acids Res.* 44.D1 (Jan. 2016), pp. D1220–8.

[129] Michael M Mysinger et al. “Directory of Useful Decoys, Enhanced (DUD-E): Better Ligands and Decoys for Better Benchmarking”. In: *Journal of Medicinal Chemistry* 55.14 (July 2012), pp. 6582–6594.

[130] John Irwin and Brian K Shoichet. “ZINC–a free database of commercially available compounds for virtual screening.” In: *Journal of Chemical Information and Modeling* 45.1 (Jan. 2005), pp. 177–182.

[131] John Irwin et al. “ZINC: A Free Tool to Discover Chemistry for Biology”. In: *Journal of Chemical Information and Modeling* 52.7 (June 2012), pp. 1757–1768.

[132] Noel T Southall and Ajay. “Kinase Patent Space Visualization Using Chemical Replacements”. In: *Journal of Medicinal Chemistry* 49.6 (Mar. 2006), pp. 2103–2109.

[133] Yi-Hui Peng et al. “Protein Kinase Inhibitor Design by Targeting the Asp-Phe-Gly (DFG) Motif: The Role of the DFG Motif in the Design of Epidermal Growth Factor Receptor Inhibitors”. In: *Journal of Medicinal Chemistry* 56.10 (May 2013), pp. 3889–3903.

[134] Peng Wu, Thomas E Nielsen, and Mads H Clausen. “FDA-approved small-molecule kinase inhibitors”. In: *Trends in Pharmacological Sciences* 36.7 (July 2015), pp. 422–439.

[135] Ye Hu and Jürgen Bajorath. “Exploring the Scaffold Universe of Kinase Inhibitors”. In: *Journal of Medicinal Chemistry* 58.1 (Jan. 2015), pp. 315–332.

[136] Giuseppe La Regina et al. “New Indole Tubulin Assembly Inhibitors Cause Stable Arrest of Mitotic Progression, Enhanced Stimulation of Natural Killer Cell Cytotoxic Activity, and Repression of Hedgehog-Dependent Cancer”. In: *Journal of Medicinal Chemistry* 58.15 (Aug. 2015), pp. 5789–5807.

[137] WHB Sauer and M K Schwarz. “Molecular shape diversity of combinatorial libraries: A prerequisite for broad bioactivity”. In: *Journal of Chemical Information and Computer Sciences* 43.3 (2003), pp. 987–1003.

[138] Kai Sommer et al. “UNICON: A Powerful and Easy-to-Use Compound Library Converter”. In: *Journal of Chemical Information and Modeling* 56.6 (June 2016), pp. 1105–1111.

[139] David Hoksza et al. “Molpher: a software framework for systematic chemical space exploration.” In: *Journal of cheminformatics* 6.1 (2014), p. 7.

[140] Karen T Schomburg, Lars Wetzer, and Matthias Rarey. “Interactive design of generic chemical patterns”. In: *Drug Discovery Today* 18.13-14 (July 2013), pp. 651–658.

[141] Nicholas C Firth. “Computation of ECFP Tanimoto Similarities with Pipeline Pilot (FAK1)”. University College London. Personal communication, Oct. 2015.

[142] Oliv Eidam. “Computation of ECFP Tanimoto Similarities with Pipeline Pilot (AKT2)”. Roche Pharma Research and Early Development, Roche Innovation Center Basel. Personal communication, Oct. 2015.

A
Tools

A.1 Fragment Space Commander

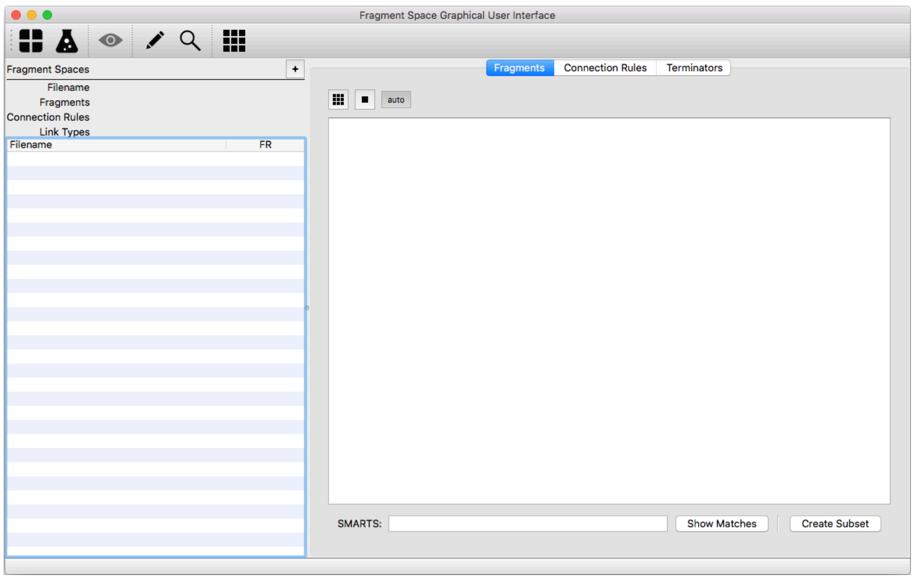
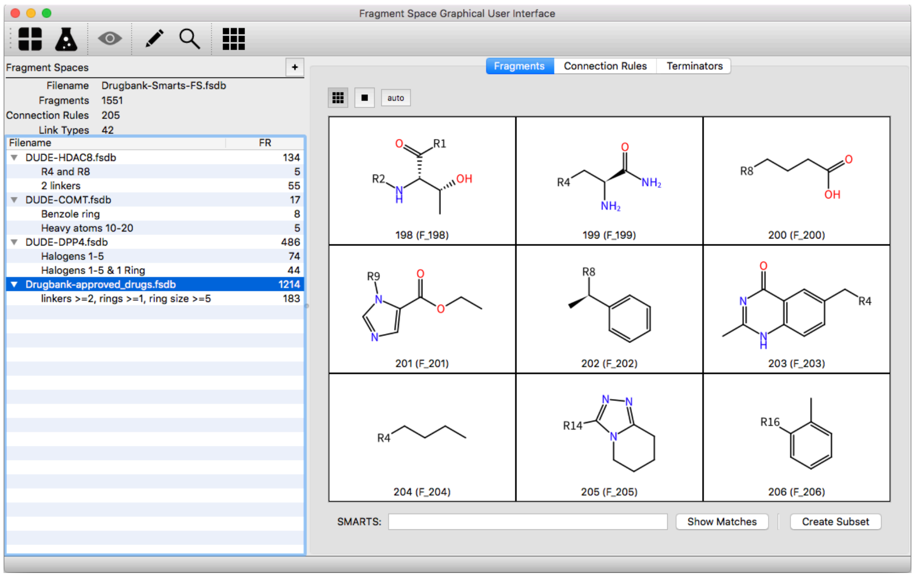


Figure A.1: Fragment Space Commander windows after startup.



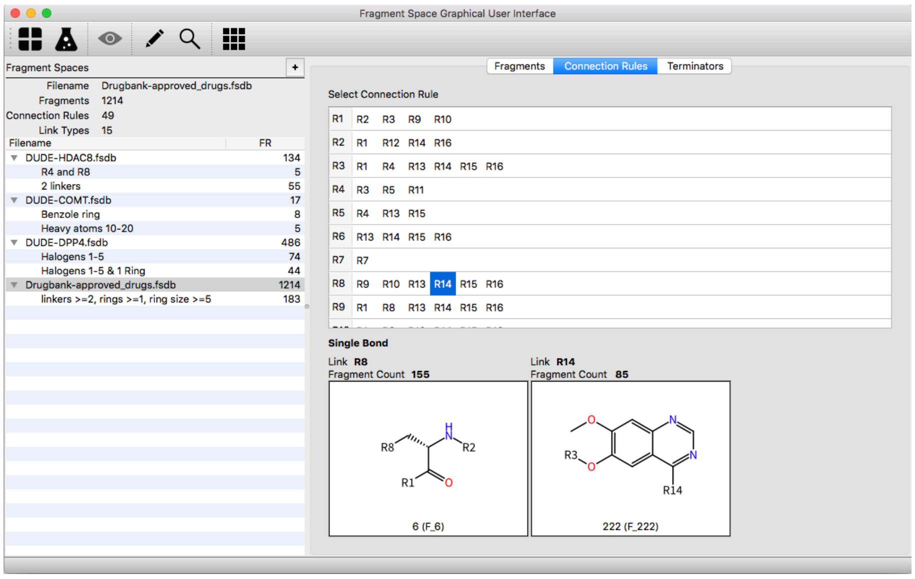
Fragment Space Graphical User Interface

Fragment Spaces

Filename	Drugbank-Smarts-FS.fsdb
Fragments	1551
Connection Rules	205
Link Types	42

Filename	FR
DUDE-HDAC8.fsdb	134
R4 and R8	5
2 linkers	55
DUDE-COMT.fsdb	17
Benzole ring	8
Heavy atoms 10-20	5
DUDE-DPP4.fsdb	486
Halogens 1-5	74
Halogens 1-5 & 1 Ring	44
Drugbank-approved_drugs.fsdb	1214
linkers >=2, rings >=1, ring size >=5	183

Figure A.2: View mode of Fragment Space Commander with selected fragment view.



Fragment Space Graphical User Interface

Fragment Spaces

Filename	Drugbank-approved_drugs.fsdb
Fragments	1214
Connection Rules	49
Link Types	15

Filename	FR
DUDE-HDAC8.fsdb	134
R4 and R8	5
2 linkers	55
DUDE-COMT.fsdb	17
Benzole ring	8
Heavy atoms 10-20	5
DUDE-DPP4.fsdb	486
Halogens 1-5	74
Halogens 1-5 & 1 Ring	44
Drugbank-approved_drugs.fsdb	1214
linkers >=2, rings >=1, ring size >=5	183

Select Connection Rule

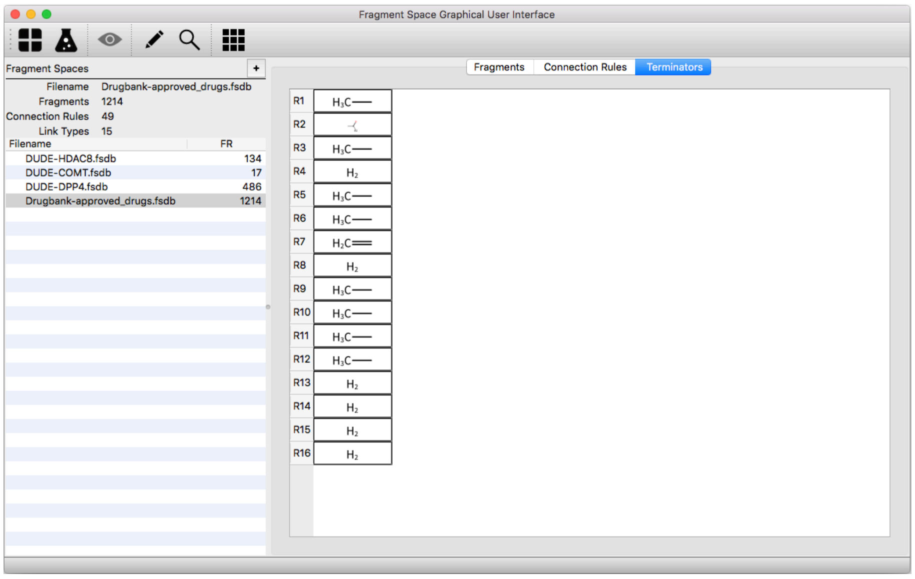
R1	R2	R3	R9	R10		
R2	R1	R12	R14	R16		
R3	R1	R4	R13	R14	R15	R16
R4	R3	R5	R11			
R5	R4	R13	R15			
R6	R13	R14	R15	R16		
R7	R7					
R8	R9	R10	R13	R14	R15	R16
R9	R1	R8	R13	R14	R15	R16

Single Bond

Link R8
Fragment Count 155

Link R14
Fragment Count 85

Figure A.3: View mode of Fragment Space Commander with selected connection rule view.

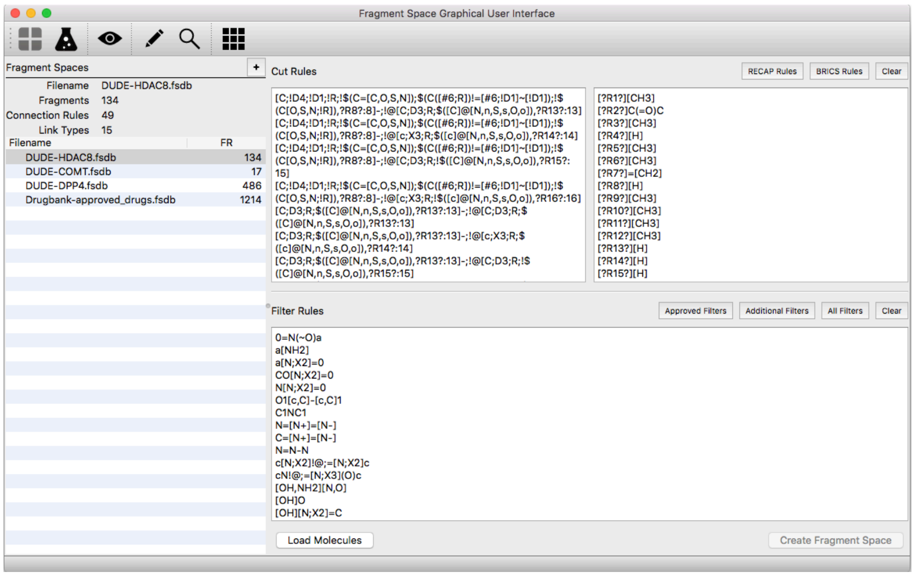


Fragment Spaces

Filename	Drugbank-approved_drugs.fsdb
Fragments	1214
Connection Rules	49
Link Types	15

Filename	FR
DUDE-HDAC8.fsdb	134
DUDE-COMT.fsdb	17
DUDE-DPP4.fsdb	486
Drugbank-approved_drugs.fsdb	1214

Figure A.4: View mode of Fragment Space Commander with selected terminator view.



Fragment Spaces

Filename	DUDE-HDAC8.fsdb
Fragments	134
Connection Rules	49
Link Types	15

Filename	FR
DUDE-HDAC8.fsdb	134
DUDE-COMT.fsdb	17
DUDE-DPP4.fsdb	486
Drugbank-approved_drugs.fsdb	1214

Cut Rules

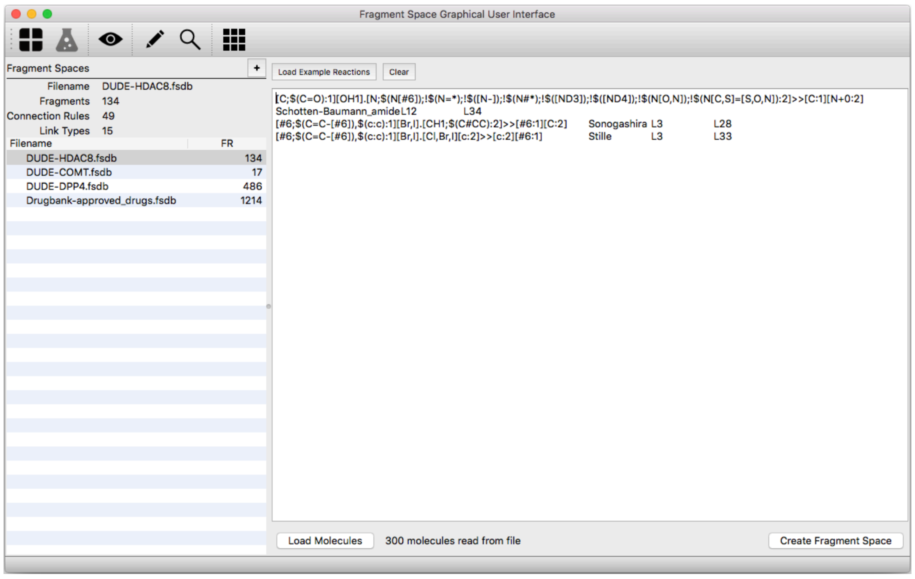
```
[C;!D4;!D1;!R;!$(C=[C,O,S,N]);$(C([#6;R])!=[#6;!D1]~[!D1]);!$
(C[O,S,N;!R]),?R8?:8]-;!@[C;D3;R;$([C]@[N,n,S,s,O,o]),?R13?:13]
[C;!D4;!D1;!R;!$(C=[C,O,S,N]);$(C([#6;R])!=[#6;!D1]~[!D1]);!$
(C[O,S,N;!R]),?R8?:8]-;!@[c;X3;R;$([c]@[N,n,S,s,O,o]),?R14?:14]
[C;!D4;!D1;!R;!$(C=[C,O,S,N]);$(C([#6;R])!=[#6;!D1]~[!D1]);!$
(C[O,S,N;!R]),?R8?:8]-;!@[C;D3;R;!$([C]@[N,n,S,s,O,o]),?R15?:
15]
[C;!D4;!D1;!R;!$(C=[C,O,S,N]);$(C([#6;R])!=[#6;!D1]~[!D1]);!$
(C[O,S,N;!R]),?R8?:8]-;!@[c;X3;R;!$([c]@[N,n,S,s,O,o]),?R16?:16]
[C;D3;R;$([C]@[N,n,S,s,O,o]),?R13?:13]-;!@[C;D3;R;$
([C]@[N,n,S,s,O,o]),?R13?:13]
[C;D3;R;$([C]@[N,n,S,s,O,o]),?R13?:13]-;!@[c;X3;R;$
([c]@[N,n,S,s,O,o]),?R14?:14]
[C;D3;R;$([C]@[N,n,S,s,O,o]),?R13?:13]-;!@[C;D3;R;!$
([C]@[N,n,S,s,O,o]),?R15?:15]
```

```
[?R1?][CH3]
[?R2?]C(=O)C
[?R3?][CH3]
[?R4?][H]
[?R5?][CH3]
[?R6?][CH3]
[?R7?]=[CH2]
[?R8?][H]
[?R9?][CH3]
[?R10?][CH3]
[?R11?][CH3]
[?R12?][CH3]
[?R13?][H]
[?R14?][H]
[?R15?][H]
```

Filter Rules

```
0=N(~O)a
a[NH2]
a[N;X2]=0
CO[N;X2]=0
N[N;X2]=0
O1[c,C]-[c,C]1
C1NC1
N=[N+]=[N-]
C=[N+]=[N-]
N=N-N
c[N;X2]!@;=[N;X2]c
cN!@;=[N;X3](O)c
[OH,NH2][N,O]
[OH]O
[OH][N;X2]=C
```

Figure A.5: Fragmentation view of Fragment Space Commander.



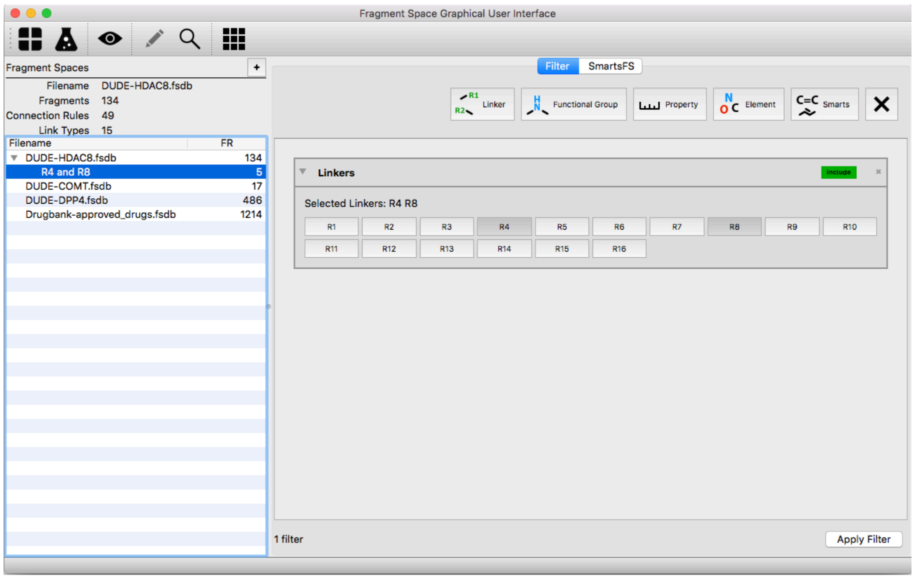
Filename	DUDE-HDAC8.fsdb
Fragments	134
Connection Rules	49
Link Types	15

Filename	FR
DUDE-HDAC8.fsdb	134
DUDE-COMT.fsdb	17
DUDE-DPP4.fsdb	486
Drugbank-approved_drugs.fsdb	1214

```
[C;$(C=O):1][OH1].[N;$(N[#6]);!$(N=*);!$([N-]);!$(N#*);!$([ND3]);!$([ND4]);!$(N[O,N]);!$(N[C,S]=[S,O,N]):2]>>[C:1][N+0:2]
Schotten-Baumann_amideL12          L34
[#6;$(C=C-[#6]),$(c:c):1][Br,I].[CH1;$(C#CC):2]>>[#6:1][C:2]      Sonogashira L3          L28
[#6;$(C=C-[#6]),$(c:c):1][Br,I].[Cl,Br,I][c:2]>>[c:2][#6:1]         Stille      L3          L33
```

300 molecules read from file

Figure A.6: Reaction-based construction view of Fragment Space Commander.



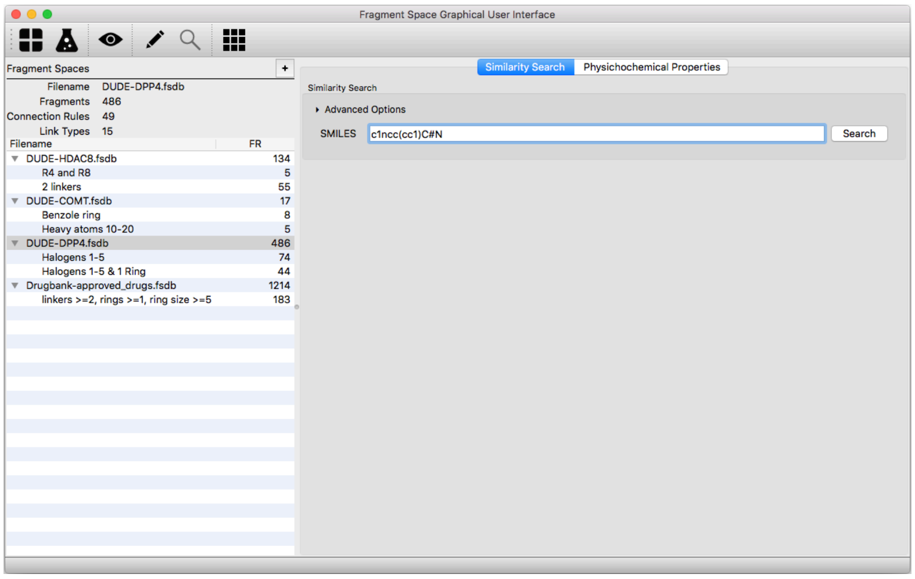
Filename	DUDE-HDAC8.fsdb
Fragments	134
Connection Rules	49
Link Types	15

Filename	FR
DUDE-HDAC8.fsdb	134
R4 and R8	5
DUDE-COMT.fsdb	17
DUDE-DPP4.fsdb	486
Drugbank-approved_drugs.fsdb	1214

Selected Linkers: R4 R8

1 filter

Figure A.7: Edit mode of Fragment Space Commander with selected filter view.


Fragment Space Graphical User Interface
Similarity Search
Physicochemical Properties
Advanced Options
Search

Fragment Spaces

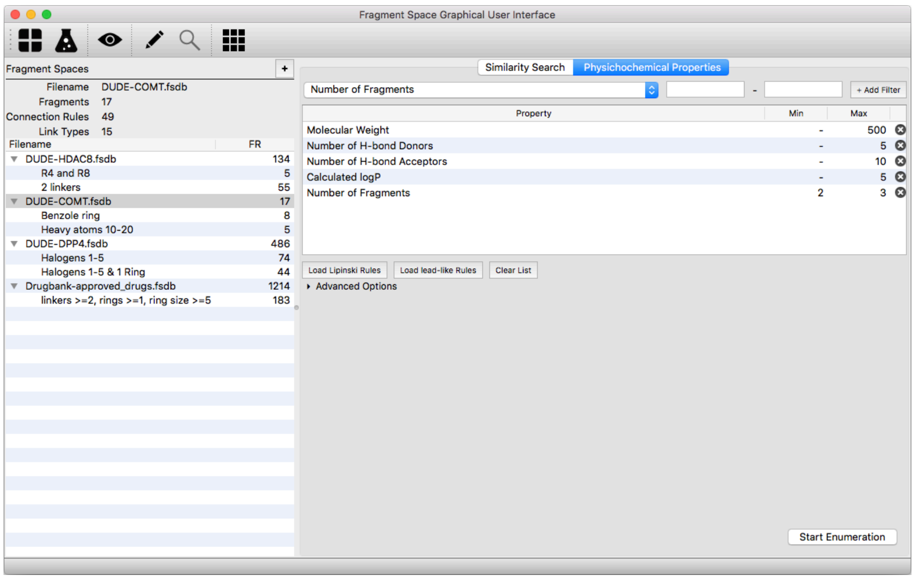
Filename	DUDE-DPP4.fsdb
Fragments	486
Connection Rules	49
Link Types	15

Filename	FR
▼ DUDE-HDAC8.fsdb	134
R4 and R8	5
2 linkers	55
▼ DUDE-COMT.fsdb	17
Benzole ring	8
Heavy atoms 10-20	5
▼ DUDE-DPP4.fsdb	486
Halogens 1-5	74
Halogens 1-5 & 1 Ring	44
▼ Drugbank-approved_drugs.fsdb	1214
linkers >=2, rings >=1, ring size >=5	183

Similarity Search

SMILES c1ncc(cc1)C#N

Figure A.8: Search mode of Fragment Space Commander with selected similarity search view.



Fragment Spaces

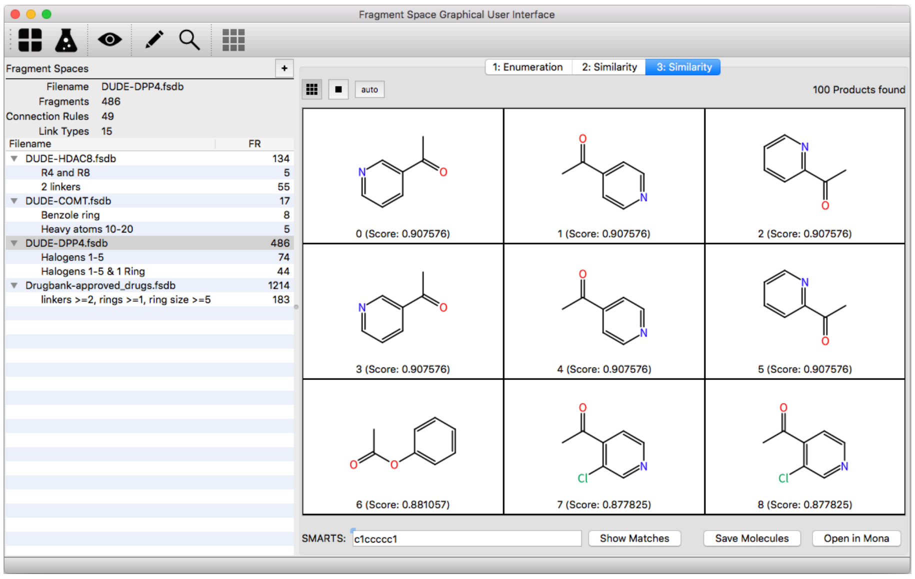
Filename	DUDE-COMT.fsdb
Fragments	17
Connection Rules	49
Link Types	15

Filename	FR
▼ DUDE-HDAC8.fsdb	134
R4 and R8	5
2 linkers	55
▼ DUDE-COMT.fsdb	17
Benzole ring	8
Heavy atoms 10-20	5
▼ DUDE-DPP4.fsdb	486
Halogens 1-5	74
Halogens 1-5 & 1 Ring	44
▼ Drugbank-approved_drugs.fsdb	1214
linkers >=2, rings >=1, ring size >=5	183

Number of Fragments

Property	Min	Max
Molecular Weight	-	500
Number of H-bond Donors	-	5
Number of H-bond Acceptors	-	10
Calculated logP	-	5
Number of Fragments	2	3

Figure A.9: Search mode of Fragment Space Commander with selected enumeration view.



Filename	DUDE-DPP4.fsdb
Fragments	486
Connection Rules	49
Link Types	15

Filename	FR
DUDE-HDAC8.fsdb	134
R4 and R8	5
2 linkers	55
DUDE-COMT.fsdb	17
Benzole ring	8
Heavy atoms 10-20	5
DUDE-DPP4.fsdb	486
Halogens 1-5	74
Halogens 1-5 & 1 Ring	44
Drugbank-approved_drugs.fsdb	1214
linkers >=2, rings >=1, ring size >=5	183

100 Products found

0 (Score: 0.907576)
1 (Score: 0.907576)
2 (Score: 0.907576)
3 (Score: 0.907576)
4 (Score: 0.907576)
5 (Score: 0.907576)
6 (Score: 0.881057)
7 (Score: 0.877825)
8 (Score: 0.877825)

SMARTS: c1ccccc1

Figure A.10: Results view of Fragment Space Commander showing molecules from a similarity search.

A.2 FSees

```
=====================================================================
             Fragment Space Exhaustive Enumeration System
=====================================================================

This tool enumerates all molecules of a given Fragment Space according
to the connection rules of the space and constraints provided by the
user as command line parameters.

 General options:
  -h [ --help ]                   Prints help message
  -v [ --verbosity ] arg         Set verbosity level
                                 (0 = Quiet, 1 = Errors, 2 = Warnings, 3 =
                                 Info)

 Input/Output parameters:
  -i [ --input ] arg             Input Fragment Space (*.fsf or *.fsdb)
  -o [ --output ] arg            File for generated molecules in SMILES format
  -m [ --memory ] arg (=1024)    Total amount of memory in MB for database
                                 cache. Default: 1024
  -f [ --filter ] arg (=1024)    Total amount of memory in MB for Bloom filter.
                                 Default: 1024
  -t [ --tolerance ] arg (=0)    Number of properties that may not be
                                 fulfilled. Default: 0
                                 Number of fragments is never excluded.
  -c [ --initialFragments ] arg  Comma seperated list of fragment ids.
                                 Corresponding fragments are used as starting
                                 points for the enumeration. Default: None

 Properties of generated molecules as ranges (e.g. 1.2:5.6, 2:, :5.4):
  --nofFragments arg (=:5)       Number of Fragments
  --nofAtoms arg                 Number of Atoms
  --molWeight arg                Molecular Weight
  --nofNonHydrogenAtoms arg      Number of Non-Hydrogen Atoms
  --nofNonHydrogenBonds arg      Number of Non-Hydrogen Bonds
  --nofRings arg                 Number of Rings
  --nofHBondAcceptors arg        Number of H-bond Acceptors
  --nofLipinskiAcceptors arg     Number of H-bond Acceptors as defined in the
                                 Lipinski rule-of-5
  --nofHBondDonors arg           Number of H-bond Donors
  --nofLipinskiDonors arg        Number of H-bond Donors as defined in the
                                 Lipinski rule-of-5
  --volume arg                   Volume
  --nofRotatableBonds arg        Number of Rotatable Bonds
  --nofStereoCenters arg         Number of Stereo Centers
  --logP arg                     Calculated logP
  --tpsa arg                     Topological Polar Surface Area
 Example - Lipinski's rule of five
  fragspace_enumerator -i FRAGSPACE.fsf -o output.smi --molWeight :500 \
  --nofHBondAcceptors :10 --nofHBondDonors :5 --logP :5.0 -t 1
```

A.3 fragspace_reaction

```
======================================================================
                Reaction-based Fragment Space Creator
======================================================================

This tool creates a fragment space based on synthetic reactions and
a list of molecules.

 Available options:

 General options:
  -h [ --help ]           Prints help message
  -v [ --verbosity ] arg  Set verbosity level
                          (0 = Quiet, 1 = Errors, 2 = Warnings, 3 = Info)

 Input options:
  -m [ --molecules ] arg  List of molecules (suffix is required)
  -r [ --reactions ] arg  List of reactions as SMIRKS

 Output options:
  -o [ --output ] arg     fragment space output file as SQLite database
                           e.g., *.db, *.fsdb
  -l [ --linker ] arg     linker names and corresponding reaction
                           e.g. *.txt, *.tab
  -f [ --fragments ] arg  list of fragments
```

A.4 molecule_shredder

```
======================================================================
                   ReiSSwolf - A Molecule Shredder
======================================================================

This tool shreds a set of molecules into a list of fragments according
to user specified cut rules. It then generates a fragment space from
these fragments deriving connection rules from the cut rules.

 Supported file formats for -m and -f options:
     *.mol *.mol2 *.pdb *.sdf *.smi *.smiles

 Available options:
```

```
 General options:
  -h [ --help ]                           Prints help message
  -v [ --verbosity ] arg                 Set verbosity level
                                         (0 = Quiet, 1 = Errors, 2 = Warnings, 3
                                         = Info)

 Input options:
  -m [ --molecules ] arg                 Input Molecule file (suffix is
                                         required)
  -c [ --cutRules ] arg                  Input cut rules as SMARTS pattern
                                         (*.sma)
  -t [ --terminators ] arg               Terminators file (suffix is required)

 Output options (at least one option is required):
  -o [ --output ] arg                    fragment space output file as SQLite
                                         database
                                          e.g., *.db, *.fsdb
  -f [ --fragments ] arg                 fragments-only output file (suffix is
                                         required)

 Filter options:
  -r [ --removeLargeFrags ] [=arg(=1)]  remove fragments with heavy atoms > 16
                                         or ring size > 8
  -p [ --post ] arg                      post-processing rules as SMARTS pattern
                                         (*.sma)
```

A.5 molecule_fingerprints

```
======================================================================
                           Fingerprint Tool
======================================================================

Calculates pairwise similarities between two sets of molecules.
Fingerprint Type and Similarity measure can be choosen by user.

 General options:
  -h [ --help ]             Prints help message
  -v [ --verbosity ] arg   Set verbosity level
                           (0 = Quiet, 1 = Errors, 2 = Warnings, 3 = Info)
  -j [ --jobs ] arg (=1)   Number of threads to use (0 = auto, 1 = single
                           [default]), e.g.: -j 4

 Input/Output parameters:
  -m [ --molecules ] arg   Molecule Sets (*.smi, *.mol2, *.sdf), e.g.: -m
                           file1 file2
  -o [ --output ] arg      Similarity Values

 Fingerprint parameters:
  -d [ --mode ] arg         Mode: all, best [default]
                            all: all pairwise similarities above cutoff (see
                            --cutoff)
                            best: only the highest value of similarity between
                            m_x in set X and all m_y of set Y
                            (--cutoff is ignored)
  -c [ --cutoff ] arg       Cutoff for similarity, only values above cutoff
                            will be saved.
                            Does not apply for mode best.
  -f [ --fingerprint ] arg  Fingerprint type: TT, ECFP_<0,2,4,6,8,10>,
                            FCFP_<0,2,4,6,8,10>
  -s [ --similarity ] arg   Similarity measure: Tanimoto, Cosine, Hamming,
                            Euclidian, Dice

 Example:
  Tanimoto Similarity of ECFP_4 fingerprints
  fingerprints -m set1.smi set2.smi -f ECFP_4 -s Tanimoto
```

A.6 Other Tools

A.6.1 fragspace_converter

```
 Available options:

 General options:
  -h [ --help ]           Prints help message
  -v [ --verbosity ] arg  Set verbosity level
                          (0 = Quiet, 1 = Errors, 2 = Warnings, 3 = Info)

 Input options:
  -i [ --input ] arg      Input file(s), suffix *.fsf is required.

 Output options:
  -o [ --output ] arg     Output file, suffix *.fsdb is required.
```

A.6.2 fragspace_creator

```
======================================================================
                   Fragment Space Creator
======================================================================

This tool creates a fragment space file from a list of fragments,
connections rules, and terminators.


 Available options:
```

```
 General options:
  -h [ --help ]              Prints help message
  -v [ --verbosity ] arg    Set verbosity level
                            (0 = Quiet, 1 = Errors, 2 = Warnings, 3 = Info)

 Input options:
  -f [ --fragments ] arg    fragments file (suffix is required)
  -c [ --cutRules ] arg     Input cut rules as SMARTS pattern (*.sma)
  -t [ --terminators ] arg  Terminators file (suffix is required)

 Output options:
  -o [ --output ] arg       fragment space output file as SQLite database
                             e.g., *.db, *.fsdb
```

A.6.3 fragspace_info

```
======================================================================
                      Fragment Space Info Tool
======================================================================

This tool prints information about a given Fragment Space.

 General options:
  -h [ --help ]           Prints help message

 Input/Output parameters:
  -i [ --input ] arg     Input Fragment Space (*.fsf or *.fsdb)
```

A.6.4 molecule_calc_pmi

```
======================================================================
    Calculate Normalized PMI Ratios (NPR) for a list of molecules.
======================================================================

 This calculates the principle moments of inertia for a list of
 molecules and therefrom the NPR values. If 3D Coordinates are not
 supplied, coordinates are generated although this might not be the
 lowest energy conformation.

 Valid file types are: *.mol *.mol2 *.sdf *.smi *.smiles

 Input options:
  -i [ --molecules ] arg  Input Molecule file (suffix is required)
  -m [ --mode ] arg (=1)  Mode: (1) Use coordinates from file (default), (2)
                          generate simple coordinates, (3) use ensemble from
                          conformation generator
  -j [ --jobs ] arg (=1)  Number of threads to use (0 = auto, 1 = single
                          [default]), e.g.: -j 4
```

```
General options:
 -h [ --help ]            Prints help message
 -v [ --verbosity ] arg  Set verbosity level
                         (0 = Quiet, 1 = Errors, 2 = Warnings, 3 = Info)
```

B Additional Experimental Results

B.1 ECFP Validation

The NAOMI-ECFP implementation was validated based on the reference implementation in Pipeline Pilot[23]. Molecules from two classes of the DUD-E dataset [129] were used: FAK1 and AKT2 with 100 and 117 molecules respectively. For each class, an all-by-all similarity comparison of ECFP_6 was carried out using Tanimoto similarity. The results of Pipeline Pilot [141, 142] and NAOMI are compared in Figure B.1. The correlation is very good. Small deviations arise from the differences in the underlying molecule models, i.e., how certain properties are assigned to atoms. Unfortunately, this is not consistent across cheminformatics frameworks. For instance, many file formats do not include hydrogens so that these must be added on initialization of a molecule instance. Depending on the automatic assignment this may lead to different protonation states and charges of a certain atom. Since ECFP values in this project were all created with NAOMI and only compared amongst themselves this has no effect on the results presented.

[23]http://accelrys.com

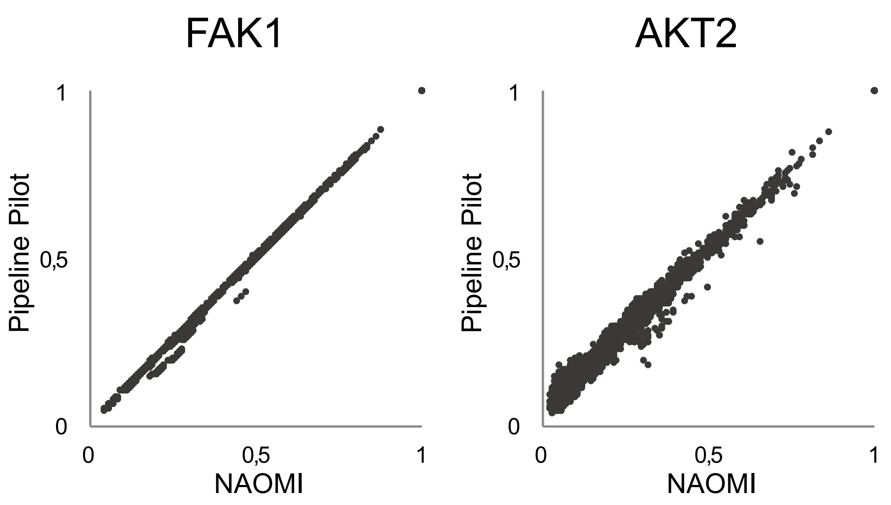


Figure B.1: Scatterplots of similarity values computed with Pipeline Pilot and NAOMI based on ECFP_6 and Tanimoto coefficient. For each DUD-E class, an all-by-all comparison of the molecules was carried out.

B.2 DUDE Enumeration

Table B.1: List of DUD-E Targets

Target Name	PDB	Molecules	Description
AA2AR	3eml	482	Adenosine A2a receptor
ABL1	2hzi	182	Tyrosine-protein kinase ABL
ACE	3bkl	282	Angiotensin-converting enzyme
ACES	1e66	453	Acetylcholinesterase
ADA	2e1w	93	Adenosine deaminase
ADA17	2oi0	532	ADAM17
ADRB1	2vt4	247	Beta-1 adrenergic receptor
ADRB2	3ny8	231	Beta-2 adrenergic receptor
AKT1	3cqw	293	Serine/threonine-protein kinase AKT
AKT2	3d0e	117	Serine/threonine-protein kinase AKT2
ALDR	2hv5	159	Aldose reductase
AMPC	1l2s	48	Beta-lactamase
ANDR	2am9	269	Androgen Receptor
AOFB	1s3b	122	Monoamine oxidase B
BACE1	3l5d	283	Beta-secretase 1
BRAF	3d4q	152	Serine/threonine-protein kinase B-raf
CAH2	1bcd	492	Carbonic anhydrase II
CASP3	2cnk	199	Caspase-3

CDK2	1h00	474	Cyclin-dependent kinase 2
COMT	3bwm	41	Catechol O-methyltransferase
CP2C9	1r9o	120	Cytochrome P450 2C9
CP3A4	3nxu	170	Cytochrome P450 3A4
CSF1R	3krj	166	Macrophage colony stimulating factor receptor
CXCR4	3odu	40	C-X-C chemokine receptor type 4
DEF	1lru	102	Peptide deformylase
DHI1	3frj	387	11-beta-hydroxysteroid dehydrogenase 1
DPP4	2i78	533	Dipeptidyl peptidase IV
DRD3	3pbl	480	Dopamine D3 receptor
DYR	3nxo	231	Dihydrofolate reductase
EGFR	2rgp	542	Epidermal growth factor receptor erbB1
ESR1	1sj0	383	Estrogen receptor alpha
ESR2	2fsz	367	Estrogen receptor beta
FA10	3kl6	537	Coagulation factor X
FA7	1w7x	114	Coagulation factor VII
FABP4	2nnq	47	Fatty acid binding protein adipocyte
FAK1	3bz3	100	Focal adhesion kinase 1
FGFR1	3c4f	139	Fibroblast growth factor receptor 1
FKB1A	1j4h	111	FK506-binding protein 1A
FNTA	3e37	592	Protein farnesyltransferase/geranylgeranyltransferase type I alpha subunit
FPPS	1zw5	85	Farnesyl diphosphate synthase
GCR	3bqd	258	Glucocorticoid receptor
GLCM	2v3f	54	Beta-glucocerebrosidase
GRIA2	3kgc	158	Glutamate receptor ionotropic, AMPA 2
GRIK1	1vso	101	Glutamate receptor ionotropic kainate 1
HDAC2	3max	185	Histone deacetylase 2
HDAC8	3f07	170	Histone deacetylase 8
HIVINT	3nf7	100	Human immunodeficiency virus type 1 integrase
HIVPR	1xl2	536	Human immunodeficiency virus type 1 protease
HIVRT	3lan	338	Human immunodeficiency virus type 1 reverse transcriptase
HMDH	3ccw	170	HMG-CoA reductase
HS90A	1uyg	88	Heat shock protein HSP 90-alpha
HXK4	3f9m	92	Hexokinase type IV
IGF1R	2oj9	148	Insulin-like growth factor I receptor
INHA	2h7l	44	Enoyl-[acyl-carrier-protein] reductase
ITAL	2ica	138	Leukocyte adhesion glycoprotein LFA-1 alpha
JAK2	3lpb	130	Tyrosine-protein kinase JAK2
KIF11	3cjo	116	Kinesin-like protein 1
KIT	3g0e	166	Stem cell growth factor receptor
KITH	2b8t	57	Thymidine kinase

KPCB	2i0e	135	Protein kinase C beta
LCK	2of2	420	Tyrosine-protein kinase LCK
LKHA4	3chp	171	Leukotriene A4 hydrolase
MAPK2	3m2w	101	MAP kinase-activated protein kinase 2
MCR	2aa2	94	Mineralocorticoid receptor
MET	3lq8	166	Hepatocyte growth factor receptor
MK01	2ojg	79	MAP kinase ERK2
MK10	2zdt	104	c-Jun N-terminal kinase 3
MK14	2qd9	578	MAP kinase p38 alpha
MMP13	830c	572	Matrix metalloproteinase 13
MP2K1	3eqh	121	Dual specificity mitogen-activated protein kinase kinase 1
NOS1	1qw6	100	Nitric-oxide synthase, brain
NRAM	1b9v	98	Neuraminidase
PA2GA	1kvo	99	Phospholipase A2 group IIA
PARP1	3l3m	508	Poly [ADP-ribose] polymerase-1
PDE5A	1udt	398	Phosphodiesterase 5A
PGH1	2oyu	195	Cyclooxygenase-1
PGH2	3ln1	435	Cyclooxygenase-2
PLK1	2owb	107	Serine/threonine-protein kinase PLK1
PNPH	3bgs	103	Purine nucleoside phosphorylase
PPARA	2p54	373	Peroxisome proliferator-activated receptor alpha
PPARD	2znp	240	Peroxisome proliferator-activated receptor delta
PPARG	2gtk	484	Peroxisome proliferator-activated receptor gamma
PRGR	3kba	293	Progesterone receptor
PTN1	2azr	130	Protein-tyrosine phosphatase 1B
PUR2	1njs	50	GAR transformylase
PYGM	1c8k	77	Muscle glycogen phosphorylase
PYRD	1d3g	111	Dihydroorotate dehydrogenase
RENI	3g6z	104	Renin
ROCK1	2etr	100	Rho-associated protein kinase 1
RXRA	1mv9	131	Retinoid X receptor alpha
SAHH	1li4	63	Adenosylhomocysteinase
SRC	3el8	524	Tyrosine-protein kinase SRC
TGFR1	3hmm	133	TGF-beta receptor type I
THB	1q4x	103	Thyroid hormone receptor beta-1
THRB	1ype	461	Thrombin
TRY1	2ayw	449	Trypsin I
TRYB1	2zec	148	Tryptase beta-1
TYSY	1syn	109	Thymidylate synthase
UROK	1sqt	162	Urokinase-type plasminogen activator
VGFR2	2p2i	409	Vascular endothelial growth factor receptor 2
WEE1	3biz	102	Serine/threonine-protein kinase WEE1

XIAP	3hl5	100	Inhibitor of apoptosis protein 3

Table B.2: Physicochemical constraints used for individual DUD-E enumerations. The values represent the upper and lower quartile of the property distribution for each property.

Target	Molecular Weight	logP	Donors	Acceptors
AA2AR	343 – 448	7 – 10	2 – 3	1.66 – 3.38
ABL1	399 – 526	6 – 8	1 – 3	4.11 – 5.67
ACE	365 – 457	6 – 8	2 – 3	1.88 – 3.39
ACES	379 – 515	4 – 6	0 – 2	3.94 – 6.35
ADA	275 – 369	6 – 7	2 – 3	1.88 – 3.13
ADA17	431 – 519	7 – 10	2 – 3	2.21 – 3.75
ADRB1	375 – 499	5 – 7	3 – 4	2.78 – 4.57
ADRB2	363 – 512	5 – 8	2 – 4	2.82 – 4.68
AKT1	398 – 508	5 – 8	2 – 4	3.72 – 5.4
AKT2	390 – 529	6 – 9	2 – 4	2.49 – 5.62
ALDR	306 – 394	5 – 7	1 – 2	2 – 3.65
AMPC	237 – 346	5 – 7	1 – 2	0.48 – 3.11
ANDR	310 – 417	3 – 5	0 – 1	3.19 – 4.92
AOFB	235 – 330	3 – 5	0 – 1	2.33 – 3.87
BACE1	479 – 566	6 – 8	3 – 4	3.41 – 5.36
BRAF	374 – 519	6 – 8	1 – 3	3.95 – 5.82
CAH2	333 – 478	6 – 10	2 – 4	1.09 – 3.09
CASP3	411 – 527	8 – 10.5	0 – 3	1.44 – 3.36
CDK2	338 – 442	6 – 8	2 – 4	2.69 – 4.2
COMT	231 – 346	6 – 8	2 – 3	1.21 – 3.19
CP2C9	356 – 522	4 – 7	1 – 2	3.68 – 5.97
CP3A4	386 – 534	5 – 8	1 – 2	3.74 – 5.13
CSF1R	386 – 483	6 – 9	1 – 3	3.32 – 4.99
CXCR4	357 – 421	4 – 5	1 – 3	3.34 – 4.72
DEF	336 – 418	6 – 7	2 – 3	1.85 – 3.31
DHI1	326 – 442	4 – 6	0 – 1	3.29 – 4.82
DPP4	338 – 421	5 – 7	2 – 2	1.19 – 2.99
DRD3	365 – 465	4 – 6	0 – 1	3.62 – 5.05
DYR	328 – 447	6 – 10	4 – 7	1.63 – 3.37
EGFR	376 – 496	6 – 8	1 – 3	3.67 – 5.15
ESR1	331 – 482	3 – 5	1 – 2	4.44 – 6.14
ESR2	314 – 476	3 – 5	1 – 2	4.06 – 6.04
FA10	479 – 548	7 – 9	2 – 5	3.09 – 4.95
FA7	430 – 529	7 – 9	4.25 – 7	2.94 – 5.41

FABP4	318 – 447	3 – 5	1 – 1.5	4.28 – 6.58
FAK1	400 – 477	7 – 9	1 – 3	3.48 – 4.26
FGFR1	390 – 511	6 – 9	2 – 3	3.09 – 5.13
FKB1A	386 – 485	5 – 7	0 – 0	3.24 – 4.76
FNTA	445 – 525	6 – 7	0 – 2	4.16 – 5.59
FPPS	268 – 309	7 – 8	5 – 5	0.02 – 0.99
GCR	392 – 489	3 – 5	1 – 2	4.62 – 6.46
GLCM	288 – 395	5 – 8	3 – 5	– 0.28 – 2.71
GRIA2	293 – 422	6 – 10	2 – 4	0.92 – 3
GRIK1	249 – 342	5 – 9	2 – 4	– 0.49 – 1.84
HDAC2	326 – 442	5 – 8	2 – 3	2.66 – 4.27
HDAC8	324 – 425	5 – 7	2 – 3	2.47 – 4.23
HIVINT	336 – 443	6 – 9	2 – 4	1.69 – 3.37
HIVPR	487 – 576	6 – 9	2 – 4	3.86 – 5.87
HIVRT	314 – 391	4 – 6	1 – 2	3.14 – 4.35
HMDH	429 – 521	5 – 8	1 – 3	4.32 – 5.46
HS90A	371 – 467	6 – 8	2 – 4	3.2 – 4.18
HXK4	369 – 465	6 – 8	1 – 2	4.09 – 4.98
IGF1R	459 – 537	7 – 9	2 – 4	3.58 – 5.54
INHA	294 – 380	3 – 4	1 – 2	4.33 – 5.24
ITAL	491 – 562	5 – 7	0 – 1	4.47 – 6.34
JAK2	347 – 464	7 – 8	1 – 2	2.96 – 4.42
KIF11	361 – 455	4 – 6	0 – 3	3.62 – 4.62
KIT	400 – 497	6 – 9	2 – 4	3.88 – 5.23
KITH	343 – 496	7 – 9	3 – 3	1.13 – 2.76
KPCB	406 – 512	6 – 8	1 – 3	3.21 – 4.52
LCK	401 – 520	6 – 9	2 – 3	3.98 – 5.54
LKHA4	313 – 440	3 – 7	0 – 2	3.43 – 4.92
MAPK2	320 – 434	5 – 7	2 – 4	2.42 – 4.09
MCR	379 – 438	4 – 6	1 – 2	4.24 – 5.52
MET	427 – 528	7 – 9	1 – 2	3.57 – 5.68
MK01	374 – 475	6 – 7	3 – 4	3.66 – 4.93
MK10	367 – 441	6 – 7	1 – 3	3.18 – 4.74
MK14	386 – 502	5 – 8	1 – 3	4.03 – 5.69
MMP13	430 – 516	7 – 10	2 – 3	2.25 – 4.26
MP2K1	383 – 510	5 – 8	1 – 4	3.75 – 5.61
NOS1	207 – 370	3 – 5	2 – 3	1.81 – 3.89
NRAM	308 – 366	6 – 8	4 – 6	– 0.13 – 1.06
PA2GA	400 – 491	6 – 7	1.5 – 3	2.98 – 4.58
PARP1	298 – 421	5 – 7	1 – 3	1.78 – 3.27
PDE5A	404 – 497	6 – 10	1 – 1	2.8 – 4.45
PGH1	300 – 375	3 – 5	0 – 2	3.67 – 5.11
PGH2	331 – 415	3 – 5	0 – 2	3.9 – 5.23

PLK1	410 – 510	6 – 8	2 – 3	3.5 – 5.66
PNPH	246 – 292	5 – 8	4 – 5	0.01 – 1.86
PPARA	438 – 511	5 – 7	1 – 1	5.09 – 6.49
PPARD	440 – 522	5 – 6	1 – 1	5.47 – 6.68
PPARG	426 – 507	5 – 7	1 – 1	4.93 – 6.5
PRGR	310 – 420	3 – 4	0 – 1	4.26 – 6.02
PTN1	439 – 557	5 – 8	1 – 3	3.84 – 6.84
PUR2	449 – 478	11 – 11	7 – 8	1.03 – 1.62
PYGM	364 – 462	6 – 8	2 – 3	3.06 – 4.58
PYRD	355 – 414	4 – 5	1 – 2	4.12 – 5.64
RENI	496 – 577	6 – 9	1 – 5	3.04 – 5.64
ROCK1	311 – 390	5 – 8	2 – 3	2.06 – 3.85
RXRA	365 – 435	2 – 5	1 – 1	5.6 – 6.9
SAHH	247 – 282	7 – 9	4 – 4	– 1.19 – – 0.1
SRC	439 – 531	6 – 9	1 – 3	3.98 – 5.9
TGFR1	319 – 419	5 – 7	1 – 2	3.68 – 4.66
THB	408 – 494	4 – 7	2 – 3	4 – 5.48
THRB	447 – 539	8 – 10	4 – 6	1.27 – 3.23
TRY1	443 – 538	7 – 10	4 – 6	1.5 – 3.96
TRYB1	443 – 542	8 – 10	2 – 5	1.6 – 3.53
TYSY	427 – 533	7 – 12	3 – 5	1.57 – 3.74
UROK	344 – 460	5 – 9	4 – 5	2.46 – 5.09
VGFR2	401 – 504	6 – 9	1 – 3	3.8 – 5.65
WEE1	425 – 520	6 – 8	2 – 3	3.84 – 5.24
XIAP	442 – 534	7 – 9	3 – 5	1.44 – 3.16

Table B.3: List of DUD-E targets and corresponding ChEMBL target ID and the number of unique molecules in this set.

Target	ChEMBL ID	# Molecules
AA2AR	CHEMBL251	6822
ABL1	CHEMBL1862	2870
ACE	CHEMBL1808	685
ACES	CHEMBL4780	232
ADA	CHEMBL2966	444
ADA17	CHEMBL3706	1311
ADRB1	-	
ADRB2	CHEMBL210	4243
AKT1	CHEMBL262	4708
AKT2	CHEMBL2431	2053
ALDR	CHEMBL1900	697

AMPC	CHEMBL2026	61868
ANDR	CHEMBL1871	2804
AOFB	CHEMBL2039	2188
BACE1	CHEMBL4822	4245
BRAF	CHEMBL5145	1153
CAH2	CHEMBL205	5239
CASP3	CHEMBL2334	2393
CDK2	CHEMBL301	3444
COMT	CHEMBL2023	79
CP2C9	CHEMBL3397	21294
CP3A4	CHEMBL340	25976
CSF1R	CHEMBL1844	2174
CXCR4	CHEMBL2107	539
DEF	CHEMBL4976	2
DHI1	CHEMBL4235	2186
DPP4	CHEMBL284	3747
DRD3	CHEMBL234	4169
DYR	CHEMBL202	1273
EGFR	CHEMBL203	7810
ESR1	CHEMBL206	5003
ESR2	CHEMBL242	3723
FA10	CHEMBL244	5756
FA7	CHEMBL3991	457
FABP4	CHEMBL2083	102
FAK1	CHEMBL2695	1609
FGFR1	CHEMBL3650	2902
FKB1A	CHEMBL1902	546
FNTA	CHEMBL271	1
FPPS	CHEMBL1782	217
GCR	CHEMBL2034	3544
GLCM	CHEMBL2179	11847
GRIA2	CHEMBL3503	95
GRIK1	CHEMBL2919	113
HDAC2	CHEMBL1937	585
HDAC8	CHEMBL3192	959
HIVINT	-	
HIVPR	-	
HIVRT	-	
HMDH	CHEMBL402	1187
HS90A	CHEMBL3880	1488
HXK4	CHEMBL3820	1037
IGF1R	CHEMBL1957	3166
INHA	-	

ITAL	CHEMBL1803	123
JAK2	CHEMBL2971	2898
KIF11	CHEMBL4581	913
KIT	CHEMBL1936	2082
KITH	CHEMBL1075130	15
KPCB	CHEMBL3045	1079
LCK	CHEMBL258	4674
LKHA4	CHEMBL4618	497
MAPK2	CHEMBL2208	2751
MCR	CHEMBL1994	879
MET	CHEMBL3717	4069
MK01	CHEMBL4040	16384
MK10	CHEMBL2637	1639
MK14	CHEMBL260	5807
MMP13	CHEMBL280	2075
MP2K1	CHEMBL3587	1817
NOS1	CHEMBL3048	1517
NRAM	CHEMBL3377	167
PA2GA	CHEMBL3474	473
PARP1	CHEMBL3105	1243
PDE5A	CHEMBL1827	2364
PGH1	CHEMBL2949	2329
PGH2	CHEMBL4321	940
PLK1	CHEMBL3024	2398
PNPH	CHEMBL4338	267
PPARA	CHEMBL239	2929
PPARD	CHEMBL3979	2290
PPARG	CHEMBL235	4460
PRGR	CHEMBL208	1941
PTN1	CHEMBL335	3521
PUR2	CHEMBL3972	104
PYGM	CHEMBL4696	651
PYRD	CHEMBL1966	743
RENI	CHEMBL286	2780
ROCK1	CHEMBL3231	2198
RXRA	CHEMBL2061	1119
SAHH	CHEMBL2664	200
SRC	CHEMBL3655	322
TGFR1	CHEMBL4439	984
THB	CHEMBL1947	6070
THRB	CHEMBL204	7095
TRY1	CHEMBL3769	854
TRYB1	CHEMBL2617	207

TYSY	CHEMBL2555	64
UROK	CHEMBL3286	1197
VGFR2	CHEMBL279	7249
WEE1	CHEMBL5491	424
XIAP	CHEMBL4198	875

Table B.4: Enumeration Results for all 102 DUD-E Targets. The enumerations marked DNF (did not finish) show in brackets how many fragments had been processed when the process was terminated.

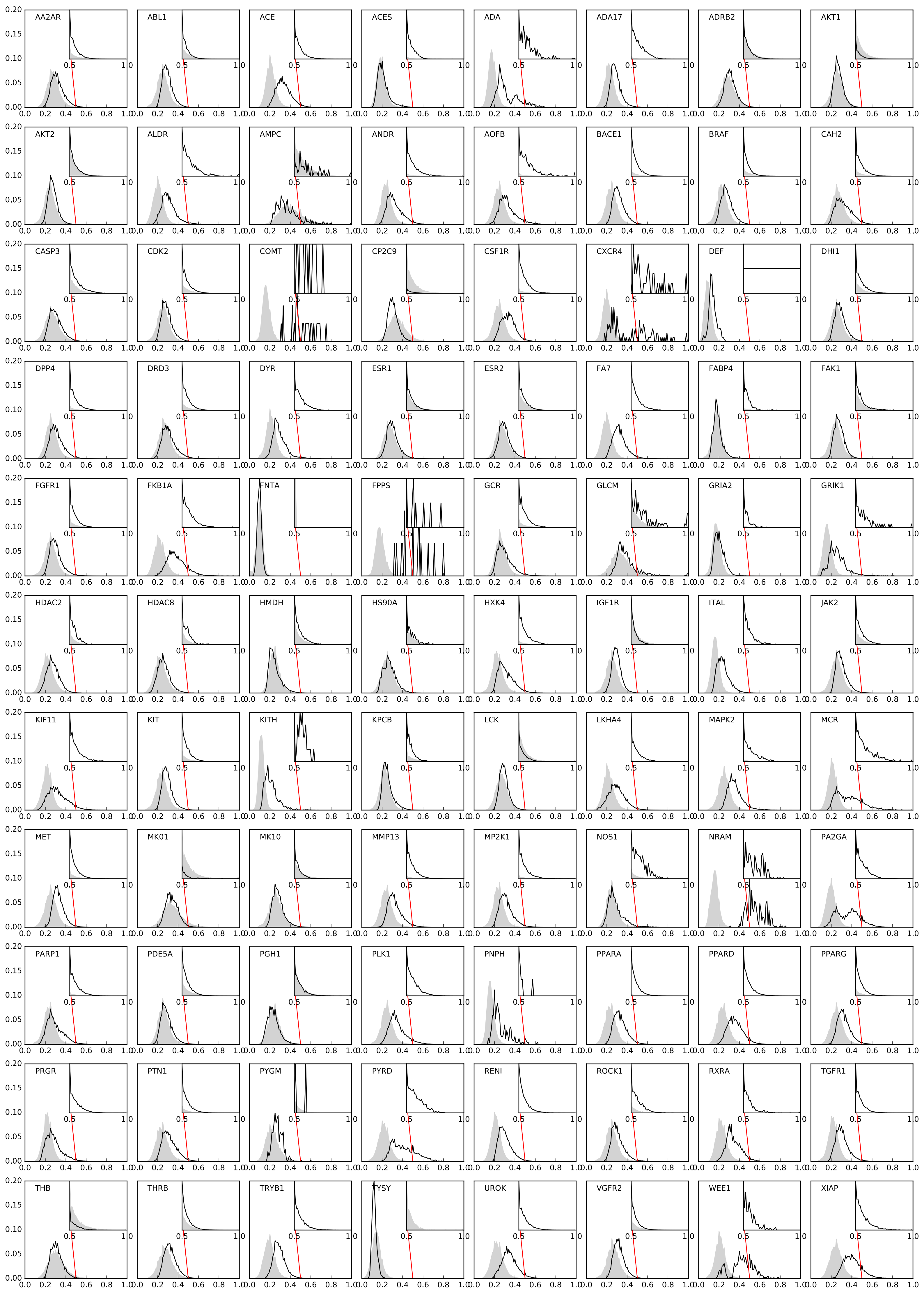
Target	Fragments	Molecules	Time [h:m:s]	Rate [Mol/s]	Redundant
AA2AR	360	16,934,058	61:35:39	85	40,275,428
ABL1	217	13,016,906	12:25:31	97	12,654,599
ACE	248	1,228,793	1:23:06	353	2,517,936
ACES	314	10,141,931	7:22:04	92	14,616,811
ADA	96	3,240	0:02:28	66	1,609
ADA17	322	6,393,032	17:04:37	54	5,940,002
ADRB1	273	2,104,870	9:55:27	44	4,983,250
ADRB2	270	8,896,562	31:08:05	235	12,899,482
AKT1	273	54,693,813	99:35:26	286	51,266,881
AKT2	138	16,432,334	27:45:30	416	4,597,172
ALDR	132	83,076	0:03:49	316	51,824
AMPC	53	489	0:00:34	11	2
ANDR	217	443,056	0:21:06	327	645,276
AOFB	123	18,565	0:02:12	71	46,758
BACE1	235	4,380,710	19:34:10	123	5,828,697
BRAF	164	24,275,229	58:45:09	265	31,974,434
CAH2	276	9,593,829	33:39:49	221	21,195,511
CASP3	194	1,152,554	2:44:29	314	807,003
CDK2	406	49,568,201	112:18:38	357	46,218,093
COMT	17	27	0:00:26	1	3
CP2C9	196	19,675,689	53:18:28	378	19,496,096
CP3A4	295	56,606,000	DNF(45%)	168	
CSF1R	184	5,252,607	22:34:40	412	4,550,848
CXCR4	30	113	0:01:07	3	159
DEF	85	2,878	0:03:38	20	2,060
DHI1	321	6,406,879	32:38:03	251	12,304,094
DPP4	506	10,566,795	17:20:46	194	15,476,815
DRD3	430	13,484,611	17:42:04	242	16,763,265
DYR	159	332,070	0:33:26	193	375,414
EGFR	461	117,733,000	DNF(10%)	195	

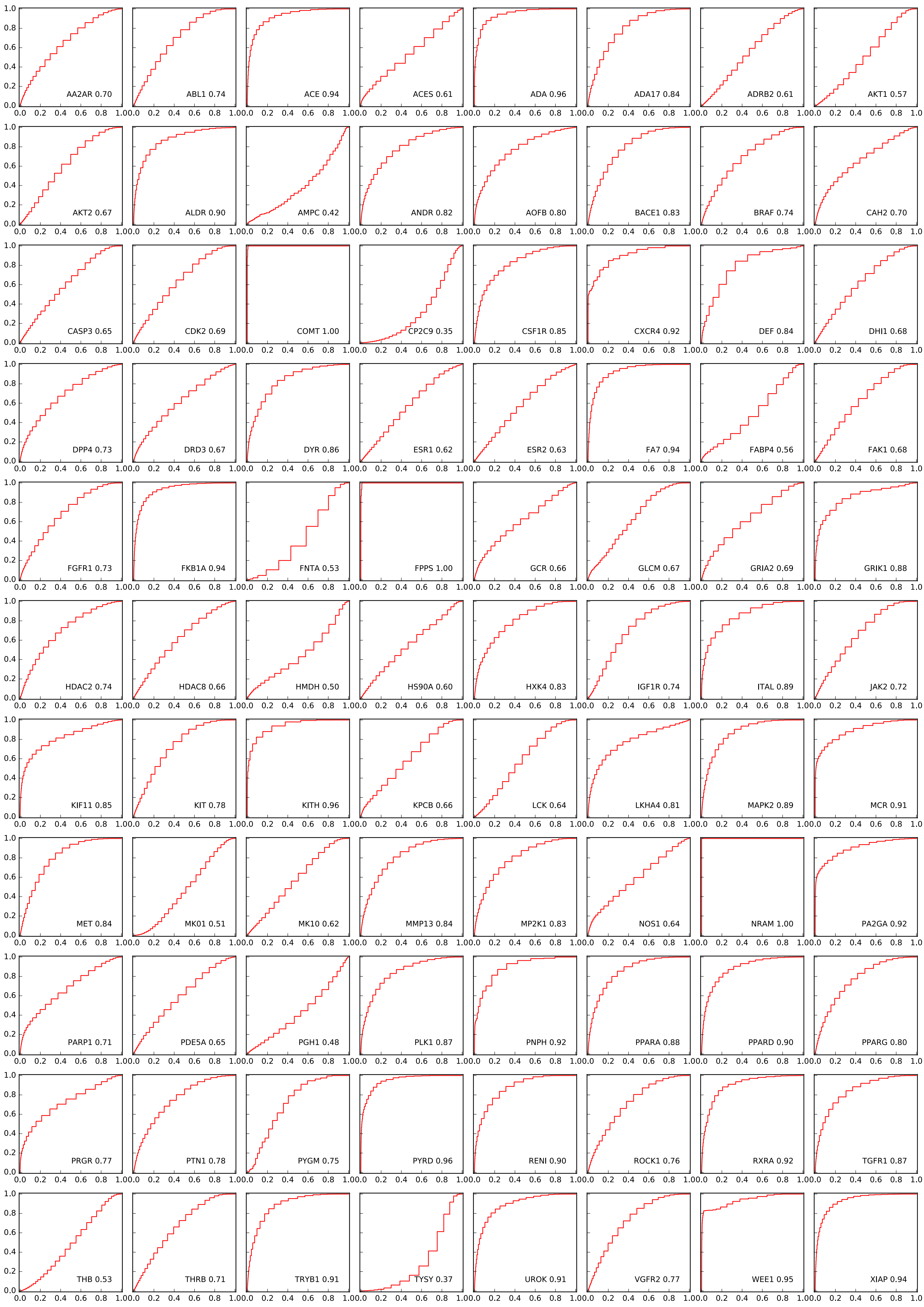
ESR1	282	51,166,308	89:44:07	158	122,564,595
ESR2	277	35,767,722	116:07:51	107	70,354,152
FA10	438	56,914,000	DNF(17%)	94	
FA7	123	648,424	0:42:07	188	925,506
FABP4	50	378,851	0:13:55	356	245,043
FAK1	128	449,719	0:25:23	226	655,374
FGFR1	156	4,004,127	2:37:10	194	3,938,795
FKB1A	89	58,326	0:03:07	237	51,846
FNTA	445	76,310,547	129:47:46	163	104,799,787
FPPS	38	15	0:00:34	0	23
GCR	170	461,344	0:26:38	228	804,574
GLCM	55	1,526	0:00:48	16	1,592
GRIA2	156	476,172	0:22:28	289	965,921
GRIK1	78	3,098	0:01:07	25	1,674
HDAC2	147	294,359	0:21:25	185	400,932
HDAC8	135	344,212	0:28:22	160	383,341
HIVINT	78	13,297	0:02:05	118	50,888
HIVPR	467	41,855,000	DNF(13%)	69	
HIVRT	291	1,643,228	0:59:52	415	1,631,167
HMDH	164	7,155,310	20:38:42	69	10,724,561
HS90A	74	82,490	0:03:25	489	55,420
HXK4	88	284,224	0:10:07	346	331,272
IGF1R	182	6,552,487	9:37:28	60	6,196,609
INHA	43	1,473	0:00:41	21	241
ITAL	132	615,852	0:34:57	221	488,496
JAK2	136	1,032,168	0:47:32	286	1,826,275
KIF11	95	48,997	0:02:56	230	25,374
KIT	182	3,556,245	3:40:30	272	5,095,650
KITH	50	4,488	0:01:46	69	2,714
KPCB	130	1,268,162	1:18:55	214	1,680,782
LCK	335	63,836,639	114:41:30	177	67,527,021
LKHA4	147	1,022,403	0:55:39	165	1,489,258
MAPK2	75	42,284	0:02:29	205	49,522
MCR	84	27,561	0:01:52	151	7,436
MET	176	5,482,021	6:02:26	494	3,959,301
MK01	80	23,706	0:02:38	99	14,644
MK10	148	7,027,819	13:41:19	114	8,157,594
MK14	450	45,650,000	DNF(2%)	75	
MMP13	338	7,612,905	14:44:31	103	7,358,221
MP2K1	117	694,273	0:20:35	405	436,397
NOS1	81	29,918	0:02:24	151	50,009
NRAM	73	371	0:01:08	4	1,089
PA2GA	91	154,990	0:16:10	168	216,815

PARP1	359	8,804,693	9:11:15	105	13,241,795
PDE5A	313	27,481,391	42:08:34	212	32,783,734
PGH1	228	17,221,670	12:02:14	306	24,910,688
PGH2	402	99,803,000	DNF(6%)	165	
PLK1	137	562,693	0:23:56	291	319,576
PNPH	83	251	0:01:12	3	990
PPARA	394	80,387,488	150:07:39	149	71,362,698
PPARD	281	10,073,914	21:19:43	76	10,144,790
PPARG	433	83,068,500	150:07:48	159	85,729,872
PRGR	217	2,880,427	1:58:41	145	4,389,914
PTN1	159	9,759,166	6:26:34	339	7,268,369
PUR2	16	0	0:00:15	0	0
PYGM	52	446	0:00:47	12	93
PYRD	60	8,532	0:01:35	112	21,191
RENI	160	1,205,371	0:46:17	361	350,516
ROCK1	111	238,033	0:20:08	204	427,179
RXRA	107	26,902	0:03:18	252	11,702
SAHH	52	2	0:00:40	0	0
SRC	400	108,955,000	DNF(21%)	180	
TGFR1	143	4,839,797	6:47:22	55	11,909,977
THB	120	176,715	0:13:12	163	259,853
THRB	423	23,477,718	41:08:46	165	29,824,465
TRY1	473	65,197,000	DNF(15%)	108	
TRYB1	150	826,988	1:17:51	295	569,181
TYSY	82	40,800	0:05:27	140	15,634
UROK	168	969,043	0:59:23	285	1,531,109
VGFR2	350	113,625,393	82:21:12	235	87,702,744
WEE1	56	712	0:00:55	7	557
XIAP	76	65,396	0:04:47	202	54,804

Figure B.2: Similarity Distribution of DUD-E datasets: Enumerated (black) and ZINC drug-like (grey) vs ChEMBL bioactives. The actual figure is shown on the next page. *This figure was originally published in [1].*

Figure B.3: ROC Curves with AUC for the similarity distribution plots of B.2. The actual figure is shown on the page after the next. *This figure was originally published in [1].*


AA2AR
ABL1
ACE
ACES
ADA
ADA17
ADRB2
AKT1
AKT2
ALDR
AMPC
ANDR
AOFB
BACE1
BRAF
CAH2
CASP3
CDK2
COMT
CP2C9
CSF1R
CXCR4
DEF
DHI1
DPP4
DRD3
DYR
ESR1
ESR2
FA7
FABP4
FAK1
FGFR1
FKB1A
FNTA
FPPS
GCR
GLCM
GRIA2
GRIK1
HDAC2
HDAC8
HMDH
HS90A
HXK4
IGF1R
ITAL
JAK2
KIF11
KIT
KITH
KPCB
LCK
LKHA4
MAPK2
MCR
MET
MK01
MK10
MMP13
MP2K1
NOS1
NRAM
PA2GA
PARP1
PDE5A
PGH1
PLK1
PNPH
PPARA
PPARD
PPARG
PRGR
PTN1
PYGM
PYRD
RENI
ROCK1
RXRA
TGFR1
THB
THRB
TRYB1
TYSY
UROK
VGFR2
WEE1
XIAP
0.00
0.05
0.10
0.15
0.20
0.0
0.2
0.4
0.6
0.8
1.0
0.5
1



AA2AR 0.70
ABL1 0.74
ACE 0.94
ACES 0.61
ADA 0.96
ADA17 0.84
ADRB2 0.61
AKT1 0.57
AKT2 0.67
ALDR 0.90
AMPC 0.42
ANDR 0.82
AOFB 0.80
BACE1 0.83
BRAF 0.74
CAH2 0.70
CASP3 0.65
CDK2 0.69
COMT 1.00
CP2C9 0.35
CSF1R 0.85
CXCR4 0.92
DEF 0.84
DHI1 0.68
DPP4 0.73
DRD3 0.67
DYR 0.86
ESR1 0.62
ESR2 0.63
FA7 0.94
FABP4 0.56
FAK1 0.68
FGFR1 0.73
FKB1A 0.94
FNTA 0.53
FPPS 1.00
GCR 0.66
GLCM 0.67
GRIA2 0.69
GRIK1 0.88
HDAC2 0.74
HDAC8 0.66
HMDH 0.50
HS90A 0.60
HXK4 0.83
IGF1R 0.74
ITAL 0.89
JAK2 0.72
KIF11 0.85
KIT 0.78
KITH 0.96
KPCB 0.66
LCK 0.64
LKHA4 0.81
MAPK2 0.89
MCR 0.91
MET 0.84
MK01 0.51
MK10 0.62
MMP13 0.84
MP2K1 0.83
NOS1 0.64
NRAM 1.00
PA2GA 0.92
PARP1 0.71
PDE5A 0.65
PGH1 0.48
PLK1 0.87
PNPH 0.92
PPARA 0.88
PPARD 0.90
PPARG 0.80
PRGR 0.77
PTN1 0.78
PYGM 0.75
PYRD 0.96
RENI 0.90
ROCK1 0.76
RXRA 0.92
TGFR1 0.87
THB 0.53
THRB 0.71
TRYB1 0.91
TYSY 0.37
UROK 0.91
VGFR2 0.77
WEE1 0.95
XIAP 0.94
0.0
0.2
0.4
0.6
0.8
1.0

B.3 Hamburg Enumerated Lead-like Set

Table B.5: Enumeration results for all 183 fragments from the "Approved Drugs" fragment space. The enumerated molecules from these enumerations were merged to form HELLS.

Fragment	Time [hh:mm:ss]	Molecules	Rate [Mols/s]
1	1:27:10	2525780	482.94
33	5:34:21	2799699	139.56
55	0:30:15	280678	154.64
58	0:02:20	7289	52.06
61	20:54:32	10762708	142.98
76	20:24:58	9976646	135.74
80	0:05:13	98786	315.61
86	0:03:52	63680	274.48
90	0:09:04	83117	152.79
94	2:41:16	1870889	193.35
96	9:31:10	4775056	139.34
114	17:46:22	6135178	95.89
120	0:00:56	5984	106.86
122	0:15:30	169202	181.94
129	0:07:00	42859	102.05
133	0:42:38	160184	62.62
136	0:00:51	4394	86.16
137	9:36:24	7144159	206.57
145	0:45:32	1568085	573.97
150	1:37:01	2113897	363.15
155	2:02:36	4169602	566.83
159	0:00:15	473	31.53
171	0:00:12	348	29.00
174	14:15:37	7638213	148.79
177	2:58:16	1741981	162.86
180	0:00:01	3	3.00
189	1:34:18	1958121	346.08
195	54:59:01	22481053	113.57
213	0:09:21	60373	107.62
222	0:37:52	360373	158.61
225	0:26:48	399761	248.61
245	0:02:26	6286	43.05
257	33:18:12	10744144	89.62
263	27:16:57	10981921	111.81
281	0:02:50	12828	75.46

287	0:11:17	97682	144.29
290	0:14:17	151577	176.87
299	9:25:22	9907425	292.06
310	11:23:51	4673656	113.91
322	0:00:46	3784	82.26
334	26:04:00	10082065	107.44
337	1:08:00	1642599	402.60
352	0:49:15	211371	71.53
361	3:22:32	1473506	121.26
363	3:52:40	2822788	202.21
364	5:43:58	2036032	98.65
370	0:29:20	447782	254.42
372	4:45:17	4169602	243.59
375	3:48:06	3318127	242.45
392	0:02:35	11246	72.55
395	2:41:30	1087796	112.26
399	0:12:09	63469	87.06
401	0:04:06	7909	32.15
408	2:39:55	1226889	127.87
414	0:22:12	568539	426.83
422	0:03:56	41266	174.86
428	2:34:23	2537878	273.98
435	34:57:37	13359055	106.14
444	4:17:47	2285774	147.78
447	8:06:09	3890886	133.39
448	0:27:52	125031	74.78
451	61:25:56	29772623	134.62
463	15:45:41	9178010	161.75
464	11:41:34	5130589	121.88
473	18:06:03	8147581	125.03
489	7:40:46	3647314	131.93
496	0:10:17	62182	100.78
501	0:21:49	138438	105.76
520	10:48:21	5887598	151.35
541	14:05:31	7850050	154.74
556	9:24:17	8095577	239.11
558	1:45:08	2060404	326.63
573	1:14:27	1461030	327.07
584	8:57:33	5290754	164.04
589	17:50:15	13415834	208.92
598	0:15:07	173374	191.15
614	1:12:37	1263043	289.89
617	8:39:42	2424000	77.74

623	0:04:13	19560	77.31
625	0:00:42	1860	44.29
643	0:31:56	63952	33.38
645	0:07:35	44693	98.23
657	0:21:52	112164	85.49
658	10:47:39	4790220	123.27
675	4:10:37	1743242	115.93
679	0:57:25	193247	56.09
684	4:41:06	2008980	119.11
689	65:55:42	27583263	116.22
692	1:11:19	1194255	279.10
700	135:00:13	110132768	226.60
707	0:12:54	41734	53.92
721	4:23:18	2987386	189.10
742	0:00:26	1775	68.27
749	5:57:19	5003417	233.38
762	0:03:29	54878	262.57
769	0:12:50	21704	28.19
771	0:32:01	106729	55.56
784	0:33:33	147334	73.19
787	16:19:52	7290319	124.00
788	0:00:00	0	
789	9:08:05	5116637	155.59
797	1:44:49	1663559	264.52
803	27:33:25	35965659	362.54
808	0:15:17	127916	139.49
809	5:36:11	5291716	262.34
816	18:20:19	30407774	460.59
823	11:44:57	9978905	235.92
849	0:02:31	66991	443.65
853	1:52:59	1257835	185.55
855	0:11:56	84618	118.18
863	27:06:42	9962880	102.08
867	8:23:59	2467839	81.61
870	15:45:52	7432961	130.97
871	6:47:03	5041883	206.44
875	0:02:07	6216	48.94
880	0:03:06	33548	180.37
895	0:18:14	205865	188.18
901	132:05:55	90987009	191.33
902	7:43:17	9418509	338.83
903	0:08:01	8040	16.72
905	5:06:06	2936181	159.87

906	20:38:20	4774360	64.26
912	2:32:29	2468938	269.86
922	17:13:49	5134955	82.78
947	1:39:47	1105248	184.61
948	0:14:54	51766	57.90
970	10:00:06	8615348	239.28
978	0:00:32	1380	43.13
980	7:20:11	4034705	152.77
982	0:52:58	214288	67.43
984	0:24:14	205410	141.27
987	16:50:26	1930894	31.85
995	9:49:04	4100012	116.00
998	1:39:35	714831	119.64
1001	7:47:28	5277696	188.17
1009	0:45:19	360373	132.54
1012	0:28:10	451723	267.29
1016	2:16:22	749481	91.60
1020	0:13:42	61013	74.23
1021	0:36:43	125382	56.91
1022	0:42:33	201734	79.02
1024	1:30:01	1431530	265.05
1029	10:49:06	5404774	138.78
1034	1:30:54	618490	113.40
1038	0:12:23	28386	38.20
1047	0:09:24	27250	48.32
1049	1:10:57	682813	160.40
1053	10:02:34	2649263	73.28
1055	1:43:37	502967	80.90
1057	22:59:07	6604629	79.82
1068	0:05:50	26887	76.82
1071	0:33:30	217213	108.07
1073	0:36:18	62698	28.79
1092	4:59:53	6710588	372.96
1096	0:03:09	3444	18.22
1099	1:08:33	410815	99.88
1106	12:52:27	6034843	130.21
1110	4:12:12	2286896	151.13
1117	1:52:43	1163915	172.10
1118	0:16:07	130084	134.52
1124	0:09:43	65802	112.87
1128	0:46:26	380308	136.51
1131	0:22:31	45023	33.33
1137	9:57:08	1175497	32.81

1140	40:46:17	27435622	186.92
1144	0:32:36	459968	235.16
1150	21:44:05	7777207	99.40
1151	2:41:05	1171491	121.21
1155	3:44:37	1271482	94.34
1160	2:22:23	2158524	252.67
1161	0:31:21	134711	71.62
1165	17:27:25	13511914	215.00
1166	9:39:03	9237852	265.89
1169	0:38:48	81258	34.90
1173	15:24:41	2557426	46.10
1174	26:33:08	2161977	22.62
1177	11:40:59	6488748	154.28
1179	1:55:23	1784488	257.76
1181	9:58:24	1360706	37.90
1186	3:08:53	892796	78.78
1199	1:24:41	756420	148.87
1204	1:34:37	585571	103.15
1208	0:06:25	3895	10.12
Sum	1441:28:01	820,467,614	
Average	7:54:44	4,483,430	152.61

B.4 FragEnum

In order to validate FSees and compare the performance with the previous implementation (FragEnum), two fragment spaces (WDI 47 and WDI 48) by Pärn et al. [29] were enumerated. All enumerations were carried out on the same computer. FSees is significantly faster than FragEnum while yielding identical results. Other comparisons were not conducted.

Table B.6: Comparison of enumeration results and runtime of FragEnum [29] and FSees [1].

	WDI 47		WDI 48	
	time [mm:ss]	molecules	time [mm:ss]	molecules
FragmEnum	01:48	73,147	31:47	95,713
FSees	00:54	73,147	07:51	95,713

C Data

C.1 Fragment Space Format

Example for the Fragment Space Format file.

```
# Fragment space file for FlexNovo representing the BRICS space
# ------------------------------------------------------------------------
# (c) Joerg Degen*, Christof Gerlach#, Andrea Zaliani*, Matthias Rarey*
#     [*] ZBH Center for Bioinformatics, University of Hamburg, Germany
#     [#] Bayer Schering Pharma AG, Berlin, Germany
#
# syntax description:
# @link_types <nof links>
#    <link name 1> <link name 2> ...
#    <link name k> <link name k+1> ... <link name <nof links>>
#
# @fragment_files <nof files>
#    < frag file 1>
#    < frag file 2>
#    :
#    < frag file <nof files>>
#
# @link_terminal_groups
#    <link name 1> <group> [<atom type> <bond type> <bond length> <torsion angle>]
#    <link name 2> <group> [<atom type> <bond type> <bond length> <torsion angle>]
#    :
#    <link name <nof links>> <group> [<atom type> <bond type> <bond length> <torsion
    angle>]
#
# @link_compatibility_matrix
#    <name of> <name of> [<aatom type> <aatom type>] <bond>  <bond>  [<torsion>]
#     link 1    link 2     for link 1   for link 2    type   length     angle
#    :
# ------------------------------------------------------------------------
```

```
@link_types 16
  R1 R2 R3 R4 R5 R6 R7 R8 R9 R10 R11 R12 R13 R14 R15 R16

@fragment_files 1
  brics/brics_4k.mol2

@link_terminal_groups
# link  group                    aatom   bond  blen   torsion
  R1    terminal_groups/r1.mol2  *       1     1.507  *
  R2    terminal_groups/r2.mol2  *       am    1.337  180
  R3    terminal_groups/r3.mol2  *       1     1.429  *
  R4    terminal_groups/r4.mol2  *       1     1.090  *
  R5    terminal_groups/r5.mol2  *       1     1.398  *
  R6    terminal_groups/r6.mol2  *       1     1.507  *
  R7    terminal_groups/r7.mol2  *       2     1.316  180
  R8    terminal_groups/r8.mol2  *       1     1.090  *
  R9    terminal_groups/r9.mol2  *       1     1.465  *
  R10   terminal_groups/r10.mol2 *       1     1.465  *
  R11   terminal_groups/r11.mol2 *       1     1.815  *
  R12   terminal_groups/r12.mol2 *       1     1.816  *
  R13   terminal_groups/r13.mol2 *       1     1.080  *
  R14   terminal_groups/r14.mol2 *       1     1.080  *
  R15   terminal_groups/r15.mol2 *       1     1.080  *
  R16   terminal_groups/r16.mol2 *       1     1.080  *

@link_compatibility_matrix
# link1  link2  aatom1  aatom2   bond  blen   torsion
  R1     R2     *       *        am    1.355  180
  R1     R3     *       *        1     1.362  *
  R1     R10    *       *        am    1.355  180
  R2     R12    *       *        1     1.656  *
  R2     R14    *       *        1     1.398  *
  R2     R16    *       *        1     1.398  *
  R3     R4     *       *        1     1.452  *
  R3     R13    *       *        1     1.429  *
  R3     R14    *       *        1     1.362  *
  R3     R15    *       *        1     1.429  *
  R3     R16    *       *        1     1.362  *
  R4     R5     *       *        1     1.469  *
  R4     R11    *       *        1     1.815  *
  R5     R13    *       *        1     1.469  *
  R5     R15    *       *        1     1.469  *
  R6     R13    *       *        1     1.507  180
  R6     R14    *       *        1     1.473  180
  R6     R15    *       *        1     1.507  180
  R6     R16    *       *        1     1.473  180
  R7     R7     *       *        2     1.316  180
  R8     R9     *       *        1     1.465  *
  R8     R10    *       *        1     1.465  *
  R8     R13    *       *        1     1.507  *
  R8     R14    *       *        1     1.473  *
  R8     R15    *       *        1     1.507  *
  R8     R16    *       *        1     1.473  *
  R9     R13    *       *        1     1.465  *
  R9     R14    *       *        1     1.355  *
  R9     R15    *       *        1     1.465  *
  R9     R16    *       *        1     1.355  *
  R10    R13    *       *        1     1.465  *
  R10    R14    *       *        1     1.355  *
  R10    R15    *       *        1     1.465  *
  R10    R16    *       *        1     1.355  *
```

```
R11    R13    *      *      1     1.816  *
R11    R14    *      *      1     1.762  *
R11    R15    *      *      1     1.816  *
R11    R16    *      *      1     1.762  *
R13    R13    *      *      1     1.530  *
R13    R14    *      *      1     1.507  *
R13    R15    *      *      1     1.530  *
R13    R16    *      *      1     1.507  *
R14    R14    *      *      1     1.473  *
R14    R15    *      *      1     1.507  *
R14    R16    *      *      1     1.473  *
R15    R15    *      *      1     1.530  *
R15    R16    *      *      1     1.507  *
R16    R16    *      *      1     1.473  *
```

C.2 BRICS Rules

BRICS retrosynthetic rules as SMARTS pattern.

```
[C;!D4;!D1;!R;!$(C=[C,O,S,N]);$(C([#6;R])!=[#6;!D1]~[!D1]);!$(C[O,S,N;!R]),?R8?:8]-;!
    @[C;D3;R;$([C]@[N,n,S,s,O,o]),?R13?:13]
[C;!D4;!D1;!R;!$(C=[C,O,S,N]);$(C([#6;R])!=[#6;!D1]~[!D1]);!$(C[O,S,N;!R]),?R8?:8]-;!
    @[c;X3;R;$([c]@[N,n,S,s,O,o]),?R14?:14]
[C;!D4;!D1;!R;!$(C=[C,O,S,N]);$(C([#6;R])!=[#6;!D1]~[!D1]);!$(C[O,S,N;!R]),?R8?:8]-;!
    @[C;D3;R;!$([C]@[N,n,S,s,O,o]),?R15?:15]
[C;!D4;!D1;!R;!$(C=[C,O,S,N]);$(C([#6;R])!=[#6;!D1]~[!D1]);!$(C[O,S,N;!R]),?R8?:8]-;!
    @[c;X3;R;!$([c]@[N,n,S,s,O,o]),?R16?:16]
[C;D3;R;$([C]@[N,n,S,s,O,o]),?R13?:13]-;!@[C;D3;R;$([C]@[N,n,S,s,O,o]),?R13?:13]
[C;D3;R;$([C]@[N,n,S,s,O,o]),?R13?:13]-;!@[c;X3;R;$([c]@[N,n,S,s,O,o]),?R14?:14]
[C;D3;R;$([C]@[N,n,S,s,O,o]),?R13?:13]-;!@[C;D3;R;!$([C]@[N,n,S,s,O,o]),?R15?:15]
[C;D3;R;$([C]@[N,n,S,s,O,o]),?R13?:13]-;!@[c;X3;R;!$([c]@[N,n,S,s,O,o]),?R16?:16]
[c;X3;R;$([c]@[N,n,S,s,O,o]),?R14?:14]-;!@[c;X3;R;$([c]@[N,n,S,s,O,o]),?R14?:14]
[c;X3;R;$([c]@[N,n,S,s,O,o]),?R14?:14]-;!@[C;D3;R;!$([C]@[N,n,S,s,O,o]),?R15?:15]
[c;X3;R;$([c]@[N,n,S,s,O,o]),?R14?:14]-;!@[c;X3;R;!$([c]@[N,n,S,s,O,o]),?R16?:16]
[C;D3;R;!$([C]@[N,n,S,s,O,o]),?R15?:15]-;!@[C;D3;R;!$([C]@[N,n,S,s,O,o]),?R15?:15]
[C;D3;R;!$([C]@[N,n,S,s,O,o]),?R15?:15]-;!@[c;X3;R;!$([c]@[N,n,S,s,O,o]),?R16?:16]
[c;X3;R;!$([c]@[N,n,S,s,O,o]),?R16?:16]-;!@[c;X3;R;!$([c]@[N,n,S,s,O,o]),?R16?:16]
[C;D3;!R;$(C(=O)([#6;R])!=[#6;!D1]~[!D1]);!$(C[O,S,N;!R]),?R6?:6]-;!@[C;D3;R;$([C]@[N
    ,n,S,s,O,o]),?R13?:13]
[C;D3;!R;$(C(=O)([#6;R])!=[#6;!D1]~[!D1]);!$(C[O,S,N;!R]),?R6?:6]-;!@[c;X3;R;$([c]@[N
    ,n,S,s,O,o]),?R14?:14]
[C;D3;!R;$(C(=O)([#6;R])!=[#6;!D1]~[!D1]);!$(C[O,S,N;!R]),?R6?:6]-;!@[C;D3;R;!$([C]@[
    N,n,S,s,O,o]),?R15?:15]
[C;D3;!R;$(C(=O)([#6;R])!=[#6;!D1]~[!D1]);!$(C[O,S,N;!R]),?R6?:6]-;!@[c;X3;R;!$([c]@[
    N,n,S,s,O,o]),?R16?:16]
[#6;!D1;!$([#6]-;!@[#6]=;!@[C,O,S;!@]);$([#6][!H;!D1]!=[!H;!D1]),?R7?:7]=;!@[#6;!D1;!
    $([#6]-;!@[#6]=;!@[C,O,S;!@]);$([#6][!H;!D1]!=[!H;!D1]),?R7?:7]
[C;!D4;!D1;!R;!$(C(=O));$(C([O;D2])[#6;!D1]~[!D1]),?R4?:4]-;!@[O;D2;$(O([C;!D1;!R])
    [#6;!D1][!D1]);!$(OCO);!$(OC=*);!$(O[P,S]),?R3?:3]
[C;D3;R;$([C]@[N,n,S,s,O,o]),?R13?:13]-;!@[O;D2;$(O([#6;!D1;R])[#6;!D1;R][!D1]);!$(
    OCO);!$(OC=;!@*);!$(O[P,S]),?R3?:3]
[c;X3;R;$([c]@[N,n,S,s,O,o]),?R14?:14]-;!@[O;D2;$(O([#6;!D1;R])[#6;!D1;R][!D1]);!$(
    OCO);!$(OC=;!@*);!$(O[P,S]),?R3?:3]
[C;D3;R;!$([C]@[N,n,S,s,O,o]),?R15?:15]-;!@[O;D2;$(O([#6;!D1;R])[#6;!D1;R][!D1]);!$(
    OCO);!$(OC=;!@*);!$(O[P,S]),?R3?:3]
[c;X3;R;!$([c]@[N,n,S,s,O,o]),?R16?:16]-;!@[O;D2;$(O([#6;!D1;R])[#6;!D1;R][!D1]);!$(
    OCO);!$(OC=;!@*);!$(O[P,S]),?R3?:3]
[C;!D4;!D1;!R;$(C(=O));$(C([O;D2])[#6;!D1]!=[!D1]),?R1?:1]-;!@[O;D2;$(O(C(=O))[#6;!D1
    ][#6;!D1]);!$(OCO);!$(O[P,S]),?R3?:3]
```

```
[P,S]-;!@[O;D2;$(O([P,S])[#6;!D1][!D1]);!$(OCO);!$(OC=*),?R3?:3]
[C;!D4;!D1;!R;$(C(=O));$(C([N;!D1])[!S;!D1][!D1]),?R1?:1]-;!@[N;!D1;!$(N(C=O)(C=O));$
    (N(C(=O))[!S;!D1][!D1]),?R2?:2]
[S;X4;$(S(=O)(=O)),?R12?:12]-;!@[#7;X3;!D1;$([#7](S(=O)(=O))[#6;!D1][!D1]);!$([#7]C(=
    O)),?R2?:2]
[C;!D4;!D1;!R;$(C(=O));$(C([n])[!S;!D1][!D1]),?R1?:1]-;!@[n;!$(n@[#6](=O)),?R9?:9]
[C;!D4;!D1;!R;!$(C(=O));$(C([N;X3;!D1])[#6;!D1][!D1]),?R4?:4]-;!@[N;X3;!D1;!$(NC=;!@[
    N,O,S]);$(N([C;!D1;!R])[#6;!D1]!=[!D1]),?R5?:5]
[C;D3;R;$([C]@[N,n,S,s,O,o]),?R13?:13]-;!@[N;X3;!D1;!$(NC=;!@[N,O,S]);$(N([#6;!D1;R])
    [#6;!D1;R]!=,@[!D1]),?R5?:5]
[c;X3;R;$([c]@[N,n,S,s,O,o]),?R14?:14]-;!@[N;X3;!D1;!$(NC=;!@[N,O,S]);$(N([#6;!D1;R])
    [#6;!D1;R]!=,@[!D1]),?R2?:2]
[C;D3;R;!$([C]@[N,n,S,s,O,o]),?R15?:15]-;!@[N;X3;!D1;!$(NC=;!@[N,O,S]);$(N([#6;!D1;R
    ])[#6;!D1;R]!=,@[!D1]),?R5?:5]
[c;X3;R;!$([c]@[N,n,S,s,O,o]),?R16?:16]-;!@[N;X3;!D1;!$(NC=;!@[N,O,S]);$(N([#6;!D1;R
    ])[#6;!D1;R]!=,@[!D1]),?R2?:2]
[C;!D4;!D1;!R;!$(C=[C,O,S,N]);!$(C[O,S,N;!R]);$(C([n])!=[#6;!D1]~[!D1]),?R8?:8]-;!@[n
    ;!$(n@[#6](=O)),?R9?:9]
[C;D3;R;$([C]@[N,n,S,s,O,o]),?R13?:13]-;!@[n;!$(n@[#6](=O)),?R9?:9]
[c;X3;R;$([c]@[N,n,S,s,O,o]),?R14?:14]-;!@[n;!$(n@[#6](=O)),?R9?:9]
[C;D3;R;!$([C]@[N,n,S,s,O,o]),?R15?:15]-;!@[n;!$(n@[#6](=O)),?R9?:9]
[c;X3;R;!$([c]@[N,n,S,s,O,o]),?R16?:16]-;!@[n;!$(n@[#6](=O)),?R9?:9]
[C;!D4;!D1;!R;!$(C=[C,O,S,N]);!$(C[O,S,N;!R]);$(C([#7;R;D3])!=[#6;!D1]~[!D1]),?R8
    ?:8]-;!@[#7;R;D3;$([#7]@[#6](=O)),?R10?:10]
[C;D3;R;$([C]@[N,n,S,s,O,o]),?R13?:13]-;!@[#7;R;D3;$([#7]@[#6](=O)),?R10?:10]
[c;X3;R;$([c]@[N,n,S,s,O,o]),?R14?:14]-;!@[#7;R;D3;$([#7]@[#6](=O)),?R10?:10]
[C;D3;R;!$([C]@[N,n,S,s,O,o]),?R15?:15]-;!@[#7;R;D3;$([#7]@[#6](=O)),?R10?:10]
[c;X3;R;!$([c]@[N,n,S,s,O,o]),?R16?:16]-;!@[#7;R;D3;$([#7]@[#6](=O)),?R10?:10]
[C;!D4;!D1;!R;$(C(=O));$(C([#7;R;D3])[!S;!D1][!D1]),?R1?:1]-;!@[#7;R;D3;$([#7]@[#6](=
    O)),?R10?:10]
[C;!D4;!D1;!R;!$(C(=S));$(C([S;D2])[#6;!D1]~[!D1]),?R4?:4]-;!@[S;D2;$(S([C;!D1;!R])
    [#6;!D1][!D1]);!$(SCS);!$(SC=*);!$(S[P,S]),?R11?:11]
[C;D3;R;$([C]@[N,n,S,s,O,o]),?R13?:13]-;!@[S;D2;$(S([#6;!D1;R])[#6;!D1;R][!D1]);!$(
    SCS);!$(SC=;!@*);!$(S[P,S]),?R11?:11]
[c;X3;R;$([c]@[N,n,S,s,O,o]),?R14?:14]-;!@[S;D2;$(S([#6;!D1;R])[#6;!D1;R][!D1]);!$(
    SCS);!$(SC=;!@*);!$(S[P,S]),?R11?:11]
[C;D3;R;!$([C]@[N,n,S,s,O,o]),?R15?:15]-;!@[S;D2;$(S([#6;!D1;R])[#6;!D1;R][!D1]);!$(
    SCS);!$(SC=;!@*);!$(S[P,S]),?R11?:11]
[c;X3;R;!$([c]@[N,n,S,s,O,o]),?R16?:16]-;!@[S;D2;$(S([#6;!D1;R])[#6;!D1;R][!D1]);!$(
    SCS);!$(SC=;!@*);!$(S[P,S]),?R11?:11]
[C;!D4;!D1;!R;$(C(S=O)[#6;!D1][!D1]),?R4?:4]-;!@[S;$(S(=O)([C;!D1;!R])[#6;!D1][!D1])
    ,?R11?:11]
[C;D3;R;$([C]@[N,n,S,s,O,o]),?R13?:13]-;!@[S;$(S(=O)([#6;!D1;R])[#6;!D1;R][!D1]),?R11
    ?:11]
[c;X3;R;$([c]@[N,n,S,s,O,o]),?R14?:14]-;!@[S;$(S(=O)([#6;!D1;R])[#6;!D1;R][!D1]),?R11
    ?:11]
[C;D3;R;!$([C]@[N,n,S,s,O,o]),?R15?:15]-;!@[S;$(S(=O)([#6;!D1;R])[#6;!D1;R][!D1]),?
    R11?:11]
[c;X3;R;!$([c]@[N,n,S,s,O,o]),?R16?:16]-;!@[S;$(S(=O)([#6;!D1;R])[#6;!D1;R][!D1]),?
    R11?:11]
```

C.3 Reaction SMARTS

Reaction SMARTS from Hartenfeller et al. [96].

```
[cH1:1]1:[c:2](-[CH2:7]-[CH2:8]-[NH2:9]):[c:3]:[c:4]:[c:5]:[c:6]:1.[#6:11]-[CH1;R0
    :10]=[OD1]>>[c:1]12:[c:2](-[CH2:7]-[CH2:8]-[NH1:9]-[C:10]-2(-[#6:11])):[c:3]:[c
    :4]:[c:5]:[c:6]:1 Pictet-Spengler
```

```
[c;r6:1](-[NH1;$(N-[#6]):2]):[c;r6:3](-[NH2:4]).[#6:6]-[C;R0:5](=[OD1])-[#8;H1,$(O-[
    CH3])]>>[c:3]2:[c:1]:[n:2]:[c:5](-[#6:6]):[n:4]@2
    benzimidazole_derivatives_carboxylic-acid/ester
[c;r6:1](-[NH1;$(N-[#6]):2]):[c;r6:3](-[NH2:4]).[#6:6]-[CH1;R0:5](=[OD1])>>[c:3]2:[c
    :1]:[n:2]:[c:5](-[#6:6]):[n:4]@2 benzimidazole_derivatives_aldehyde
[c;r6:1](-[SH1:2]):[c;r6:3](-[NH2:4]).[#6:6]-[CH1;R0:5](=[OD1])>>[c:3]2:[c:1]:[s:2]:[
    c:5](-[#6:6]):[n:4]@2 benzothiazole
[c:1](-[OH1;$(Oc1ccccc1):2]):[c;r6:3](-[NH2:4]).[c:6]-[CH1;R0:5](=[OD1])>>[c:3]2:[c
    :1]:[o:2]:[c:5](-[c:6]):[n:4]@2 benzoxazole_arom-aldehyde
[c;r6:1](-[OH1:2]):[c;r6:3](-[NH2:4]).[#6:6]-[C;R0:5](=[OD1])-[OH1]>>[c:3]2:[c:1]:[o
    :2]:[c:5](-[#6:6]):[n:4]@2 benzoxazole_carboxylic-acid
[#6:6]-[C;R0:1](=[OD1])-[CH1;R0:5](-[#6:7])-[*;#17,#35,#53].[NH2:2]-[C:3]=[SD1:4]>>[c
    :1]2(-[#6:6]):[n:2]:[c:3]:[s:4][c:5]([#6:7]):2 thiazole
[c:1](-[C;$(C-c1ccccc1):2](=[OD1:3])-[OH1]):[c:4](-[NH2:5]).[N;!H0;!$(N-N);!$(N-C=N)
    ;!$(N(-C=O)-C=O):6]-[C;H1,$(C-[#6]):7]=[OD1]>>[c:4]2:[c:1]-[C:2](=[O:3])-[N:6]-[C
    :7]=[N:5]-2 Niementowski_quinazoline
[CH0;$(C-[#6]):1]#[NH0:2]>>[C:1]1=[N:2]-N-N=N-1 tetrazole_terminal
[CH0;$(C-[#6]):1]#[NH0:2].[C;A;!$(C=O):3]-[*;#17,#35,#53]>>[C:1]1=[N:2]-N(-[C:3])-N=N
    -1 tetrazole_connect_regioisomere_1
[CH0;$(C-[#6]):1]#[NH0:2].[C;A;!$(C=O):3]-[*;#17,#35,#53]>>[C:1]1=[N:2]-N=N-N-1(-[C
    :3]) tetrazole_connect_regioisomere_2
[CH0;$(C-[#6]):1]#[CH1:2].[C;H1,H2;A;!$(C=O):3]-[*;#17,#35,#53,OH1]>>[C:1]1=[C:2]-N
    (-[C:3])-N=N-1 Huisgen_Cu-catalyzed_1,4-subst
[CH0;$(C-[#6]):1]#[CH1:2].[C;H1,H2;A;!$(C=O):3]-[*;#17,#35,#53,OH1]>>[C:1]1=[C:2]-N=
    NN(-[C:3])-1 Huisgen_Ru-catalyzed_1,5_subst
[CH0;$(C-[#6]):1]#[CH0;$(C-[#6]):2].[C;H1,H2;A;!$(C=O):3]-[*;#17,#35,#53,OH1]>>[C
    :1]1=[C:2]-N=NN(-[C:3])-1 Huisgen_disubst-alkyne
[CH0;$(C-[#6]):1]#[NH0:2].[NH2:3]-[NH1:4]-[CH0;$(C-[#6]);R0:5]=[OD1]>>[N:2]1-[C:1]=[N
    :3]-[N:4]-[C:5]=1 1,2,4-triazole_acetohydrazide
[CH0;$(C-[#6]):1]#[NH0:2].[CH0;$(C-[#6]);R0:5](=[OD1])-[#8;H1,$(O-[CH3]),$(O-[CH2]-[
    CH3])]>>[N:2]1-[C:1]=N-N-[C:5]=1 1,2,4-triazole_carboxylic-acid/ester
[#6;!$([#6](-C=O)-C=O):4]-[CH0:1](=[OD1])-[C;H1&!$(C-[*;!#6])&!$(C-C(=O)O),H2:2]-[CH0
    ;R0:3](=[OD1])-[#6;!$([#6](-C=O)-C=O):5]>>[c:1]1(-[#6:4]):[c:2]:[c:3](-[#6:5]):n:
    c(-O):c(-C#N):1 3-nitrile-pyridine
[c:1](-[C;$(C-c1ccccc1):2](=[OD1:3])-[CH3:4]):[c:5](-[OH1:6]).[C;$(C1-[CH2]-[CH2]-[N,
    C]-[CH2]-[CH2]-1):7](=[OD1])>>[O:6]1-[c:5]:[c:1]-[C:2](=[OD1:3])-[C:4]-[C:7]-1
    spiro-chromanone
[#6;!$([#6](-C=O)-C=O):4]-[CH0:1](=[OD1])-[C;H1&!$(C-[*;!#6])&!$(C-C(=O)O),H2:2]-[CH0
    ;R0:3](=[OD1])-[#6;!$([#6](-C=O)-C=O):5].[NH2:6]-[N;!H0;$(N-[#6]),H2:7]>>[C
    :1]1(-[#6:4])-[C:2]=[C:3](-[#6:5])-[N:7]-[N:6]=1 pyrazole
[c;r6:1](-[C;$(C=O):6]-[OH1]):[c;r6:2]-[C;H1,$(C-C):3]=[OD1].[NH2:4]-[NH1;$(N-[#6]);!
    $(NC=[O,S,N]):5]>>[c:1]1:[c:2]-[C:3]=[N:4]-[N:5]-[C:6]-1 phthalazinone
[#6:5]-[C;R0:1](=[OD1])-[C;H1,H2:2]-[C;H1,H2:3]-[C:4](=[OD1])-[#6:6].[NH2;$(N-[C,N])
    ;!$(NC=[O,S,N]);!$(N([#6])[#6]);!$(N~N~N):7]>>[C:1]1(-[#6:5])=[C:2]-[C:3]=[C
    :4](-[#6:6])-[N:7]-1 Paal-Knorr pyrrole
[C;$(C-c1ccccc1):1](=[OD1])-[C;D3;$(C-c1ccccc1):2]~[O;D1,H1].[CH1;$(C-c):3]=[OD1]>>[C
    :1]1-N=[C:3]-[NH1]-[C:2]=1 triaryl-imidazole
[NH1;$(N-c1ccccc1):1](-[NH2])-[c:5]:[cH1:4].[C;$(C([#6])[#6]):2](=[OD1])-[CH2;$(C
    ([#6])[#6]);!$(C(C=O)C=O):3]>>[C:5]1-[N:1]-[C:2]=[C:3]-[C:4]:1 Fischer indole
[NH2;$(N-c1ccccc1):1]-[c:2]:[c:3]-[CH1:4]=[OD1].[C;$(C([#6])[#6]):6](=[OD1])-[CH2;$(C
    ([#6])[#6]);!$(C(C=O)C=O):5]>>[N:1]1-[c:2]:[c:3]-[C:4]=[C:5]-[C:6]:1 Friedlaender
     chinoline
[*;Br,I;$(*c1ccccc1)]-[c:1]:[c:2]-[OH1:3].[CH1:5]#[C;$(C-[#6]):4]>>[c:1]1:[c:2]-[O
    :3]-[C:4]=[C:5]-1 benzofuran
[*;Br,I;$(*c1ccccc1)]-[c:1]:[c:2]-[SD2:3]-[CH3].[CH1:5]#[C;$(C-[#6]):4]>>[c:1]1:[c
    :2]-[S:3]-[C:4]=[C:5]-1 benzothiophene
[*;Br,I;$(*c1ccccc1)]-[c:1]:[c:2]-[NH2:3].[CH1:5]#[C;$(C-[#6]):4]>>[c:1]1:[c:2]-[N
    :3]-[C:4]=[C:5]-1 indole
[#6:6][C:5]#[#7;D1:4].[#6:1][C:2](=[OD1:3])[OH1]>>[#6:6][c:5]1[n:4][o:3][c:2]([#6:1])
    n1 oxadiazole
[#6;$([#6]~[#6]);!$([#6]=O):2][#8;H1:3].[Cl,Br,I][#6;H2;$([#6]~[#6]):4]>>[CH2:4][O
    :3][#6:2] Williamson ether
```

```
[#6:4]-[C;H1,$([CH0](-[#6])[#6]):1]=[OD1].[N;H2,$([NH1;D2](C)C);!$(N-[#6]=[*]):3]-[C
    :5]>>[#6:4][C:1]-[N:3]-[C:5] reductive amination
[#6;H0;D3;$([#6](~[#6])~[#6]):1]B(O)O.[#6;H0;D3;$([#6](~[#6])~[#6]):2][Cl,Br,I
    ]>>[#6:2][#6:1] Suzuki
[c;H1:3]1:[c:4]:[c:5]:[c;H1:6]:[c:7]2:[nH:8]:[c:9]:[c;H1:1]:[c:2]:1:2.O=[C:10]1[#6;H2
    :11][#6;H2:12][N:13][#6;H2:14][#6;H2:15]1>>[#6;H2:12]3[#6;H1:11]=[C:10]([c:1]1:[c
    :9]:[n:8]:[c:7]2:[c:6]:[c:5]:[c:4]:[c:3]:[c:2]:1:2)[#6;H2:15][#6;H2:14][N:13]3
    piperidine_indole
[#6;$([#6]~[#6]);!$([#6]~[S,N,O,P]):1][Cl,Br,I].[Cl,Br,I][#6;$([#6]~[#6]);!$([#6]~[S,
    N,O,P]):2]>>[#6:2][#6:1] Negishi
[C;H1&$(C([#6])[#6]),H2&$(C[#6]):1][OH1].[NH1;$(N(C=O)C=O):2]>>[C:1][N:2]
    Mitsunobu_imide
[C;H1&$(C([#6])[#6]),H2&$(C[#6]):1][OH1].[OH1;$(Oc1ccccc1):2]>>[C:1][O:2]
    Mitsunobu_phenole
[C;H1&$(C([#6])[#6]),H2&$(C[#6]):1][OH1].[NH1;$(N([#6])S(=O)=O):2]>>[C:1][N:2]
    Mitsunobu_sulfonamide
[C;H1&$(C([#6])[#6]),H2&$(C[#6]):1][OH1].[#7H1:2]1~[#7:3]~[#7:4]~[#7:5]~[#6:6]~1>>[C
    :1][#7:2]1:[#7:3]:[#7:4]:[#7:5]:[#6:6]:1 Mitsunobu_tetrazole_1
[C;H1&$(C([#6])[#6]),H2&$(C[#6]):1][OH1].[#7H1:2]1~[#7:3]~[#7:4]~[#7:5]~[#6:6]~1>>[#7
    H0:2]1:[#7:3]:[#7H0:4]([C:1]):[#7:5]:[#6:6]:1 Mitsunobu_tetrazole_2
[C;H1&$(C([#6])[#6]),H2&$(C[#6]):1][OH1].[#7:2]1~[#7:3]~[#7H1:4]~[#7:5]~[#6:6]~1>>[C
    :1][#7H0:2]1:[#7:3]:[#7H0:4]:[#7:5]:[#6:6]:1 Mitsunobu_tetrazole_3
[C;H1&$(C([#6])[#6]),H2&$(C[#6]):1][OH1].[#7:2]1~[#7:3]~[#7H1
    :4]~[#7:5]~[#6:6]~1>>[#7:2]1:[#7:3]:[#7:4]([C:1]):[#7:5]:[#6:6]:1
    Mitsunobu_tetrazole_4
[#6;c,$(C(=O)O),$(C#N):3][#6;H1:2]=[#6;H2:1].[#6;$([#6]=[#6]),$(c:c):4][Cl,Br,I
    ]>>[#6:4]/[#6:1]=[#6:2]/[#6:3] Heck_terminal_vinyl
[#6;c,$(C(=O)O),$(C#N):3][#6:2]([#6:5])=[#6;H1;$([#6][#6]):1].[#6;$([#6]=[#6]),$(c:c)
    :4][Cl,Br,I]>>[#6:4][#6;H0:1]=[#6:2]([#6:5])[#6:3] Heck_non-terminal_vinyl
[#6;$(C=C-[#6]),$(c:c):1][Br,I].[Cl,Br,I][c:2]>>[c:2][#6:1] Stille
[#6:1][C:2]#[#7;D1].[Cl,Br,I][#6;$([#6]~[#6]);!$([#6]([Cl,Br,I])[Cl,Br,I]);!$([#6]=O)
    :3]>>[#6:1][C:2](=O)[#6:3] Grignard_carbonyl
[#6:1][C;H1,$([C]([#6])[#6]):2]=[OD1:3].[Cl,Br,I][#6;$([#6]~[#6]);!$([#6]([Cl,Br,I])[
    Cl,Br,I]);!$([#6]=O):4]>>[C:1][#6:2]([OH1:3])[#6:4] Grignard_alcohol
[#6;$(C=C-[#6]),$(c:c):1][Br,I].[CH1;$(C#CC):2]>>[#6:1][C:2] Sonogashira
[C;$(C=O):1][OH1].[N;$(N[#6]);!$(N=*);!$([N-]);!$(N#*);!$([ND3]);!$([ND4]);!$(N[O,N])
    ;!$(N[C,S]=[S,O,N]):2]>>[C:1][N+0:2] Schotten-Baumann_amide
[S;$(S(=O)(=O)[C,N]):1][Cl].[N;$(NC);!$(N=*);!$([N-]);!$(N#*);!$([ND3]);!$([ND4]);!$(
    N[c,O]);!$(N[C,S]=[S,O,N]):2]>>[S:1][N+0:2] sulfon_amide
[c:1]B(O)O.[nH1;+0;r5;!$(n[#6]=[O,S,N]);!$(n~n~n);!$(n~n~c~n);!$(n~c~n~n):2]>>[c:1]-[
    n:2] N-arylation_heterocycles
[#6:3]-[C;H1,$([CH0](-[#6])[#6]);!$(CC=O):1]=[OD1].[Cl,Br,I][C;H2;$(C-[#6]);!$(CC[I,
    Br]);!$(CCO[CH3]):2]>>[C:3][C:1]=[C:2] Wittig
[Cl,Br,I][c;$(c1:[c,n]:[c,n]:[c,n]:[c,n]:[c,n]:1):1].[N;$(NC)&!$(N=*)&!$([N-])&!$(N
    #*)&!$([ND3])&!$([ND4])&!$(N[c,O])&!$(N[C,S]=[S,O,N]),H2&$(Nc1:[c,n]:[c,n]:[c,n
    ]:[c,n]:[c,n]:1):2]>>[c:1][N:2] Buchwald-Hartwig
[C;$(C([#6])[#6;!$([#6]Br)]):4](=[OD1])[CH;$(C([#6])[#6]):5]Br.[#7;H2:3][C;$(C(=N)(N)
    [c,#7]):2]=[#7;H1;D1:1]>>[C:4]1=[CH0:5][NH:3][C:2]=[N:1]1 imidazole
[c;$(c1[c;$(c[C,S,N](=[OD1])[*;R0;!OH1])]cccc1):1][C;$(C(=O)[O;H1])].[c;$(c1aaccc1)
    :2][Cl,Br,I]>>[c:1]-[c:2] decarboxylative_coupling
[c;!$(c1ccccc1);$(c1[n,c]c[n,c]c[n,c]1):1][Cl,F].[N;$(NC);!$(N=*);!$([N-]);!$(N#*);!$
    ([ND3]);!$([ND4]);!$(N[c,O]);!$(N[C,S]=[S,O,N]):2]>>[c:1][N:2]
    heteroaromatic_nuc_sub
[c;$(c1c(N(~O)~O)cccc1):1][Cl,F].[N;$(NC);!$(N=*);!$([N-]);!$(N#*);!$([ND3]);!$([ND4
    ]);!$(N[c,O]);!$(N[C,S]=[S,O,N]):2]>>[c:1][N:2] nucl_sub_aromatic_ortho_nitro
[c;$(c1ccc(N(~O)~O)cc1):1][Cl,F].[N;$(NC);!$(N=*);!$([N-]);!$(N#*);!$([ND3]);!$([ND4
    ]);!$(N[c,O]);!$(N[C,S]=[S,O,N]):2]>>[c:1][N:2] nucl_sub_aromatic_para_nitro
[N;$(N-[#6]):3]=[C;$(C=O):1].[N;$(N[#6]);!$(N=*);!$([N-]);!$(N#*);!$([ND3]);!$([ND4])
    ;!$(N[O,N]);!$(N[C,S]=[S,O,N]):2]>>[N:3]-[C:1]-[N+0:2] urea
[N;$(N-[#6]):3]=[C;$(C=S):1].[N;$(N[#6]);!$(N=*);!$([N-]);!$(N#*);!$([ND3]);!$([ND4])
    ;!$(N[O,N]);!$(N[C,S]=[S,O,N]):2]>>[N:3]-[C:1]-[N+0:2] thiourea
```