A Molecular Structure Ontology for Medicinal Chemistry

by

Carmen Stephanie Chui

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Mechanical & Industrial Engineering
University of Toronto

Abstract

A Molecular Structure Ontology for Medicinal Chemistry

Carmen Stephanie Chui
Doctor of Philosophy
Graduate Department of Mechanical & Industrial Engineering
University of Toronto
2019

The field of medicinal chemistry involves the design, synthesis, and development of new drugs that can be further enhanced with the application of ontologies. This thesis is divided into three parts: first, we discuss the need for a molecular structure ontology to aid us in the task of drug design and outline a requirements-driven approach that guided the design of the Molecular Structure Ontology (MoSt). Second, we demonstrate how the ontology is verified with incidence structures found in mathematical theories, and discuss the insights gained from the verification results. Finally, we demonstrate a new approach to building molecules by exploiting graph-theoretic properties of the ontology, outline characterization theorems that correspond to this model building process, and demonstrate how the ontology can help navigate the chemical search space for molecules.

This thesis is dedicated to my parents who have given me their unconditional support, patience, encouragement, and love throughout my life.

Acknowledgements

First and foremost, I would like to thank Professor Michael Grüninger for his support and guidance over the course of my undergraduate and graduate studies. My first class with him was in my third year of undergrad where he taught the design and analysis of information systems; at the time, little did I know I would end up continuing my studies in graduate school. Michael's passion and enthusiasm to explain concepts found in ontology engineering and information engineering resulted in my interest in doing a fourth-year research thesis with him. It is through our various discussions and brainstorming sessions that I found an interest in the work done within the research community and decided to continue with graduate school for both my Masters and Doctorate degrees. I am immensely grateful for Michael's guidance throughout my graduate school journey – this thesis would not have been possible without him.

Thank you to the members on my thesis committee, Professors Mark S. Fox and L.H. Shu, for providing invaluable feedback on how I can improve my thesis over the years. As well, I would like to thank Professors Michel Dumontier and Gary Bader for their time to review and provide constructive feedback on my thesis. Additionally, I would like to thank Professor Mark Lautens and The Lautens Research Group in the Department of Chemistry at UofT for their feedback on the underlying commonsense intuitions chemists make while reviewing molecules and their structures. I am thankful for their comments and their interest in reviewing my work.

During my time as a member of the Semantic Technologies Lab, I have been privileged to work with a wonderful group of people, both past and present. Thank you to my colleagues for sharing the graduate student experience with me.

I want to further express my gratitude to the individuals that provided me with their feedback in person, through email, or through anonymous peer-review. The discussions at the KEOD 2014, FOIS 2016, WOMoCoE 2016, FOUST 2017, and FOIS 2018 conferences and workshops were especially fruitful and thought-provoking. In particular, I would like to thank the Department of Mechanical & Industrial Engineering at UofT and The International Association for Ontology and its Applications (IAOA) for providing financial support that enabled me to attend these workshops and conferences.

I would like to thank my family and friends for their confidence in me, and the values that they have instilled in me. I am extremely grateful for the tolerance that they have demonstrated with me, and I feel so fortunate for their continued support of my academic endeavours.

Finally, I would like to thank my parents for their unconditional love, support, patience, and encouragement to pursue my studies. You have always wanted the best for me and, with this thesis, I hope I have made you proud.

Contents

List of Tables

List of Figures[1]

[1] For all skeletal diagrams in this thesis, they are drawn with the LaTeX `chemfig` package (`https://ctan.org/pkg/chemfig`); additional figures are drawn manually with Omni Group's Omnigraffle tool (`https://www.omnigroup.com/omnigraffle`), unless otherwise stated and referenced.

List of Definitions[2]

[2]Where applicable, definitions for terms that are adopted from other sources are explicitly stated and referenced.

List of Theorems

List of Proofs

List of Acronyms

ASCII American Standard Code for Information Interchange

ChEBI Chemical Entities of Biological Interest

CL Common Logic

CLIF Common Logic Interchange Format

COLORE COmmon Logic Ontology Repository

CRO Contract Research Organization

CSP Constraint Satisfaction Problem

DL Description Logic

FOL First-order logic

InChI IUPAC International Chemical Identifier

IUPAC International Union of Pure and Applied Chemistry

KB Knowledge Base

LADR Library for Automated Deduction Research

MoSt MOlecular Structure Ontology

MSOL Monadic Second-Order Logic

OWL Web Ontology Language

PSL Process Specification Language

SAR Structure Activity Relationship

SMILES Simplified molecular-input line-entry system

SWRL Semantic Web Rule Language

TPTP Thousands of Problems for Theorem Provers

Chapter 1

Introduction

Intuitions of shape are often thought of as part of everyday commonsense notions – for example, "a ball is round" or "this sheet of paper has four sides." As humans, we do not have a desire to formalize such notions of shape in everyday situations. For computers, on the other hand, we need to explicitly define what 'round' means or what it means to have four sides. There are currently many approaches in existing ontological and philosophical work that attempts to formalize such shapes in various contexts (see [GB11; Rov11; HBS11; HBO13]). With this thesis, we are interested in the chemistry domain with a focus on the *topological* shape of molecules.

1.1 The Chemistry Domain

Chemistry is a scientific domain of interest for the development of ontologies due to the potential for various ontological applications. We are specifically interested in the field of *medicinal chemistry*, where chemists apply their chemistry training to process the synthesis of new pharmaceuticals [Ame15]. This domain provides a particularly rich set of shapes that can be formalized and axiomatized in the context of artificial intelligence.

Medicinal chemists are involved with the design, chemical synthesis, and development for market of pharmaceutical agents, or bio-active molecules ('drugs'). Compounds that are used as medicines are mostly organic compounds, with some inorganic and organometallic compounds that are also useful as drugs. In particular, medicinal chemistry focuses on small organic molecules, chemical identification, and the systematic and synthetic alteration of new chemical entities to make them suitable for therapeutic use [Ame15]. The synthetic and computational aspects of medicinal chemistry also involve the study of existing drugs and agents in development in relation to their biological activities and properties, such as structure-activity relationships (SAR) that describe the relationship between the chemical or structure of a molecule and its biological activity. SAR analyses determine the chemical groups responsible for evoking biological effects in organisms. Furthermore, when medicinal chemists are designing and developing new drugs, they are *implicitly* reasoning about the *structure* of these compounds; for example, they may ask, "What shapes of molecules will fit a given target?" or "What do a particular set of molecules have in common with respect to their shape?" Currently,

molecules and their structures are described using the International Union of Pure and Applied Chemistry (IUPAC) nomenclature and data structures that primarily focus on the classification of molecules, such as class-subclass relationships. Further, existing ontological approaches of describing molecules focus primarily on computational aspects of answering classification queries or rules.

We make the distinction here that our primary area of focus when we discuss 'molecular shape' is purely topological in nature. *Topology* is the mathematical study of properties of objects that are preserved under continuous deformations, such as stretching, twisting, crumpling, and bending, but not cutting or tearing [Bru00; Mun00]. For example, a circle and an ellipse are the same topologically.

With respect to small, drug-like molecules in chemistry, they present an interesting challenge from an ontological perspective: what kind of inferences can we make from the shapes of molecules? What kinds of relationships between the shapes can be made? How can shape be characterized logically? We will not focus on the *geometric* properties of molecules, such arrangement of atoms in three-dimensional space influences many aspects of a molecule: its reactivity, polarity, phase of matter, colour, magnetism and biological activity [McM15]. Instead, we examine molecular shape in a static and two-dimensional lens in this work, due to the resulting graph-theoretic notions that arose during an examination of the intended first-order models of the ontology. Furthermore, the ontology that is presented in this thesis was designed with graph- and model-theoretic notions in mind. For future work, an area of interest would be to represent molecular shape in a dynamic setting (chemical reactions) and to determine how such geometric changes and reactive processes can be represented in first-order logic.

1.2 Ontologies

The use of ontologies in knowledge representation has become increasingly prevalent in research of various domains, such as time [HP04; Hay96], manufacturing [GM03; Chu+13], web services [GHM08], biological pathways [Dem+10], military [Lac+05], and geology [RP05]. *Ontologies* are shared conceptualizations that formally define the concepts, relationships, and semantics of a given domain of discourse. As well, they can be described in languages of varying degrees of formality and expressivity. The development of automatic reasoning systems has enabled the research community to determine the validity of logical inferences of ontologies by checking the truth of entailments in a given theory or instance of a model. For this reason, we are interested in ontologies defined in the language of first-order logic since its expressiveness allows us to define complex concepts and relationships and to verify them with well-defined inference methods and tools.

An ontology is simply defined to be "an explicit specification of a conceptualization" [Gru95]. Since domains of discourse can be represented and modelled in different ways, ontologies are known to be a shared conceptualization that allows shareability and reusability within various groups in industry [GM03]. Despite the various degrees of formality between what one refers to as 'ontologies'[1], they are composed of a vocabulary of terms, and a specification of meaning of the terms. Languages for formal ontologies

[1]These include taxonomies, thesauri, data models, and other various representations.

are closely related to mathematical logic: knowledge is specified as theories in ontologies, in which semantics are based on the notion of mathematical interpretations (models of the ontology). By writing axioms in first-order logic, it allows the verification of theories through the use of (semi)automatic theorem provers that read in a computer-interpretable version of the ontology. Without explicit semantics, the inference process needs to be conducted by an engineer, a consultant, or a domain expert. If (semi)automated analysis is not needed, two domain experts will need agree on the verification or validation of a model of the ontology; however, it is often the case such experts are not available, so explicit semantics need to be defined.

1.3 The Need for An Ontology of Molecular Structure

From what we have seen in chemistry, their existing standards and nomenclature are derived from data structures created out of convenience of handling the complexities found in the IUPAC naming rules. These shortcomings are further discussed in Chapter 2, but we make note here that these nomenclature standards focus more on classification of molecules and do not provide us with any ontological insights.

Drug Discovery Since our area of focus is medicinal chemistry, the drug discovery pipeline is also of interest: a lot of time and money is invested in the search for new and effective drugs. Modern drug discovery is a complex multi-stage process that is divided into various stages: the gathering of biological information, identifying screening hits, medicinal chemistry to increase the affinity of drug-binding, clinical testing, and long-term studies before a drug can be admitted into markets. A sample road-map of the difficulties of designing drugs by Charles River Laboratories, Inc. ('Charles River', for short) is shown in Figure 1.1. Charles River is a contract research organization (CRO) based in Wilmington, Massachusetts that specializes in a variety of pre-clinical and clinical services for the pharmaceutical industry. This particular schematic was chosen to exemplify the complexity of the drug design and testing process – we can see that the path to a successful drug that is accepted by the U.S. Food and Drug Administration (FDA), Canadian Food Inspection Agency (CFIA), and Department of Health Canada ('Health Canada') is not as straightforward as chemists and organizations would like. With the work done in this thesis, we hope to aid with the Chemistry phase outlined in blue in the drug discovery roadmap, along with the Medicinal Chemistry Phase, by providing a molecular shape ontology that can be used in conjunction with automated reasoning tools used in the ontology and theorem proving communities. Furthermore, we provide an alternative way of looking at drug design through first-order model construction and theorem proving that can possibly alleviate the synthesis challenges that medicinal chemists face.

Chemical Space Existing work in cheminformatics and drug discovery discusses the notion of 'chemical space' to describe all possible organic molecules to be considered

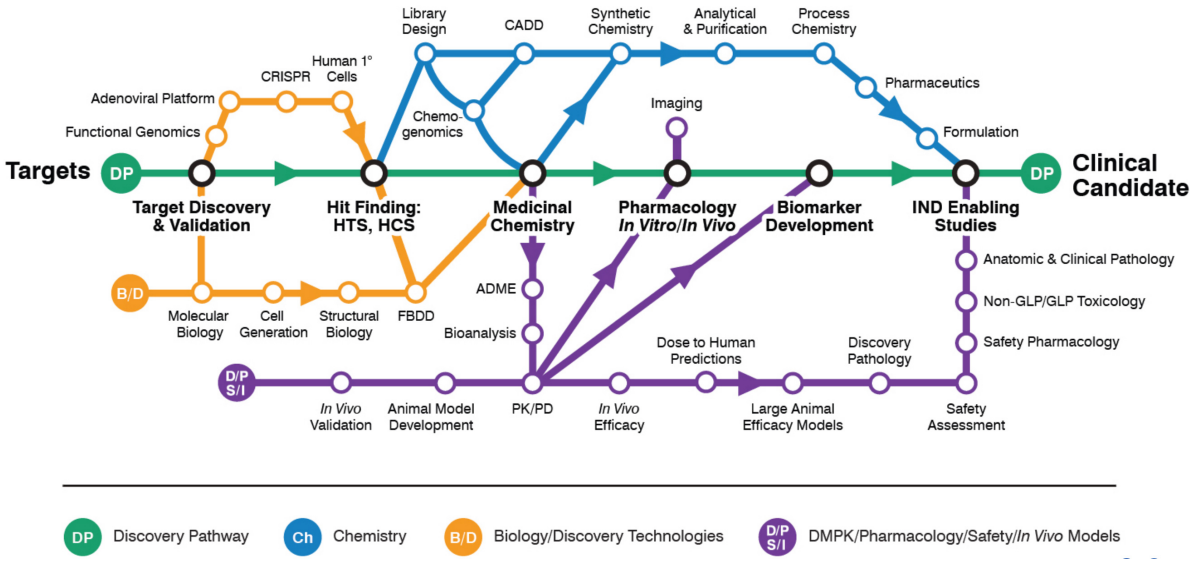


Figure 1.1: Charles River Laboratories, Inc.'s roadmap of delivering drugs. Slide #12 in [Cha17].

when searching for new drugs [RA12]. We want to provide **ontological foundations for chemical space**, where the central idea is that chemical space is characterized by the *shape* and *structure* of molecules. This means that we want to be able to represent molecules using first-order axioms and build models of these axioms. In chemistry, there is a concept of a 'chemical scaffold' that is the common *core* structure that characterizes a group of molecules in which it is contained as a substructure [Sch+07]. Structures that share a scaffold are thought to have a common synthetic pathway [STW12], so compounds in combinatorial libraries are generally grouped and based on common scaffolds. In Figure 1.2, a scaffold tree diagram for natural products (NP) is shown – we can see that various natural products can be *organized* and *categorized* according to their scaffolds. With a first-order ontology, *we want to axiomatize the molecular shapes found in the scaffold tree*: this existing notion of chemical space can be further enhanced with ontologies that describe *molecular shape*. Relating this to the scaffold tree approach, we can tailor the search in chemical space at the *knowledge* level, and constrain the search space using special techniques for first-order model construction.

1.4 Summary of Contributions

In this thesis, we discuss the intuitions of the structure of molecules and the design decisions made in the development of a molecular structure ontology. The contributions can be summarized as follows:

Formal Requirements for an Ontology of Molecular Shape We present competency questions that guide the overall design of a molecular structure ontology. These competency questions form the basis for the formalized requirements and semantic con-

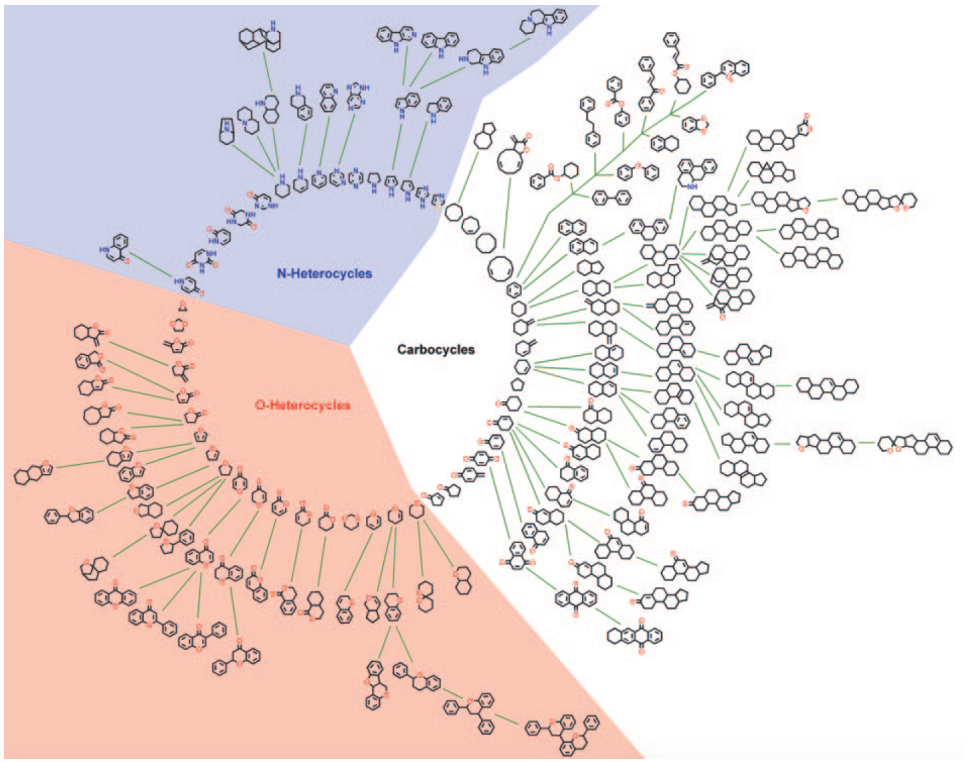


Figure 1.2: Scaffold Tree for Natural Products. The tree-like arrangement shows the structural classification and correlation of natural product scaffolds. (Figure 1 in [Koc+05]).

ditions for representing knowledge of molecular shape: molecules must be represented as graphs, components of molecules must be elements of the domain, and their permissible attachments (spiro, tether, and fusion) must be represented. Furthermore, requirements for the first-order models that result from the axiomatization are also presented.

Axiomatization for an Ontology of Molecular Shape We present an alternative approach to looking at molecular shape from commonsense intuitions of shape that is inspired by graph theory, and formally specify first-order logic axioms to describe the various components of molecular shape as the Molecular Structure Ontology (MoSt). As well, we show how these components of the ontology can be combined together via attachment relations (spiro, tether, and fusion), along with a discussion on how they can serve as building blocks for new molecules.

Verification of MoSt As part of the ontology design process, it is important to verify the axioms of the ontology to ensure the actual models of the ontology are definably equivalent to the intended models of the ontology. To do this, we discuss and utilize necessary theories from incidence geometry to verify MoSt, and discuss how the verification results provide us with insight on how models of the ontology can be used.

Techniques for Decomposing and (Re-)Composing Molecules We present techniques for the decomposition and (re)composition of molecules with theorems that de-

scribe how primitive functional groups and skeletons can be attached to each other. We also discuss algorithms for decomposing the underlying molecular graph into a molecule's building blocks, which we call primitive functional groups, and how they can be pieced together to form new molecules.

Techniques for Drug Discovery via Model Construction We present a model-theoretic approach that can be used to discover new molecules via model construction and model attachment operators. We also introduce the notion of attachment graphs that can be used with the algorithms mentioned earlier to build models of MoSt. As well, we map these attachment operators to graph-theoretic operators to show how molecules are models of MoSt.

Model-Theoretic Search Techniques We present searching techniques, framed in the context of constraint satisfaction problems (CSPs), that can be used with the ontology to discover molecules via model construction. Depending on how a query is constructed, the chemical search space can be further constrained with extensions of MoSt.

1.5 Thesis Overview

We first discuss the motivations for this work from the study of polyhedra and development of general shape ontologies, and then provide an overview of the existing approaches to describe molecular shape and a discussion on the shortcomings of these existing approaches in Chapter 2. Then, in Chapter 3, we discuss and outline the requirements needed for an ontology of molecular structure. We then partition the thesis into **three (3)** major parts, as follows:

Part I. The Molecular Structure Ontology (MoSt) provides an overview of the basic components of molecules and how they are axiomatized in the ontology:

- In Chapter 4, we provide an overview of the ontology and the process by which we developed the ontological commitments.
- In Chapter 5, we discuss the components of chemical graph and how they can be axiomatized in first-order logic.
- In Chapter 6, we discuss the notions of rings, chains, and functional groups found in chemistry.
- In Chapter 7, we discuss how the above components can be attached together.
- In Chapter 8, we discuss how bridged compounds are represented in the ontology.
- In Chapter 9, we discuss how cycles in a chemical graph can be identified as chemical rings.
- In Chapter 10, we discuss how everything can be put together to describe the skeletal framework found in molecules.

Part II. Verification of MoSt provides a discussion on the methodology used to verify the ontology with mathematical theories, along with the technical results:

- In Chapter 11, we provide a discussion on how to group axioms together into theories, and present the theories of MoSt.
- In Chapter 12, we provide an overview of ontology verification techniques, a discussion on incidence theories and their graph-theoretic concepts.
- In Chapter 13, we provide the technical results of verifying the T_{most_graph} and T_{most_group} theories, and discuss how the verification process provides us with further insight on the ontology.

Part III. Applications of MoSt provides a discussion the various applications of the ontology:

- In Chapter 14, we revisit the original ontology requirements, and discuss how building models of MoSt corresponds to designing new molecules, and what this means for medicinal chemists.
- In Chapter 15, we present a discussion of how to navigate chemical space using the ontology, along with model generation.

Finally, in Chapter 16, we present a summary of the major contributions, along with the lessons learned, and discuss possible directions for future work.

Chapter 2

Existing Approaches to Represent Shapes of Molecules[1]

In this chapter, we discuss the existing approaches found in the chemistry and ontology communities to represent molecular shape and discuss their shortcomings.

2.1 Shapes of Physical Objects & Shape Ontologies

In this section, we discuss how the shapes of physical objects plays a role in our approach to axiomatize molecular shape. We discuss notions of topological shape, polyhedra and nets, and existing shape ontologies.

2.1.1 Topological Shape

As briefly mentioned in Chapter 1, we concern ourselves with the notion of *topological shape* in the context of molecules. In mathematics, *topology* is the study of properties of space that are preserved under continuous deformations; for example, in Figure 2.1, a circle and ellipse have the same topological shape: you can squish a circle into an ellipse and stretch an ellipse into a circle, and similarly for squares and rectangles.

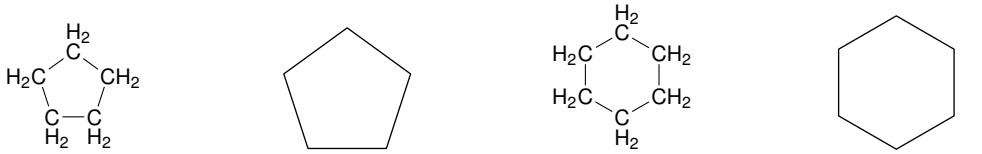
(a) A circle. **(b)** An ellipse. **(c)** A square. **(d)** A rectangle.

Figure 2.1: In topology, a circle is equivalent to an ellipse, and a square is equivalent to a rectangle.

With respect to topological shape, we are interested in looking at molecules from a topological perspective: from observation, we can think of molecules in terms of polyhedra – if we look at the skeletal formulae for molecules, we can see polygonal shapes in the

[1]This chapter extends the discussion previously published in [CG16].

formulae. For example, in Figure 2.2, cyclopentane resembles a pentagon with five sides, and cyclohexane resembles a hexagon with six sides.



(a) Cyclopentane **(b)** Pentagon **(c)** Cyclohexane **(d)** Hexagon

Figure 2.2: Molecules and polygons.

Next, we discuss polyhedra and polyhedral nets, and how they also motivate the work done in this dissertation.

2.1.2 Polyhedra & Nets

In geometry, a *polyhedron* is a three-dimensional solid with flat polygonal faces, edges, and vertices. Examples of polyhedra include cubes, prisms, and pyramids. Euler's Theorem states that the number of faces (F), vertices (V), and edges (E) are in the same way for any polyhedron:

$$\text{Faces} + \text{Vertices} = \text{Edges} + 2$$

$$F + V = E + 2$$

There are five *regular* polyhedra where all of the faces are congruent regular polygons, as shown in Figure 2.3:

1. *Regular Tetrahedrons* have 4 faces that are all equilateral triangles.
2. *Cubes* have 6 faces that are all squares.
3. *Regular Octahedrons* have 8 faces that are all equilateral triangles.
4. *Regular Dodecahedrons* have 12 faces that are all regular pentagons.
5. *Regular Icosahedrons* have 20 faces that are all equilateral triangles.

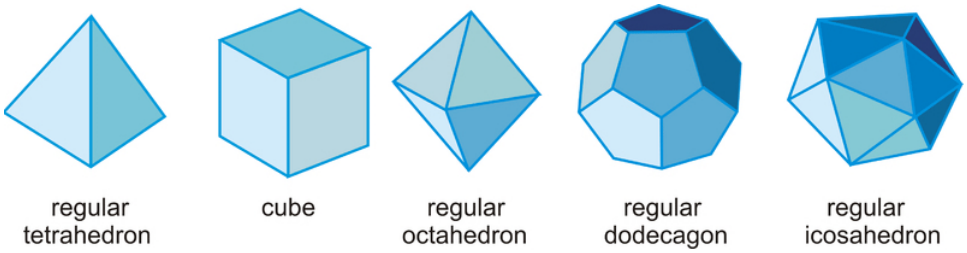


Figure 2.3: Examples of polyhedra. (Image from [CK116])

What makes polyhedra extremely interesting is when they are 'unfolded' into a *net*. These nets show the arrangement of edge-joined polygons in a plane, where these polygons

can be folded along the edges to form the faces of the polyhedron. Many different nets can exist for a given polyhedron; for example, there are eleven (11) nets [BP98] for a cube, as shown in Figure 2.4.

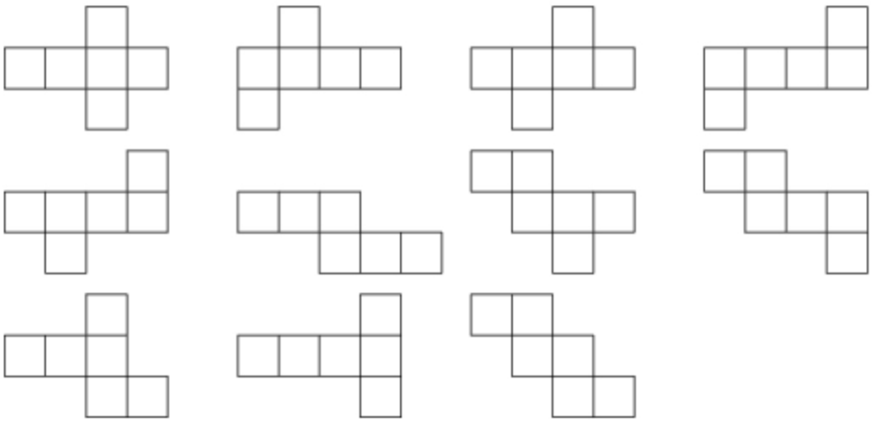
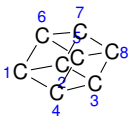


Figure 2.4: Polyhedra nets for a cube.

With these polyhedral nets, we can see that the nets can be considered as graphs since they contain vertices and edges. In fact, in algebraic topology, these polyhedral nets are considered *1-skeletons*, which are topological graphs [MS02]. Consider the multidimensional aspects of polyhedra:

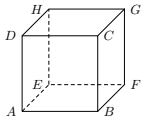
- Each face of a polyhedron is bounded by edges that are in a cycle, and
- Every edge intersects two unique points in a polyhedron.

Since these polyhedra have properties of graphs that can be generalized with commonsense notions of shape, we want to do the same with molecules: we want to generalize our understanding of shape with respect to molecules. Consider the cubane (C_8H_8) molecule in Figure 2.5: when you look at the skeletal diagram without the hydrogens in Figure 2.5(b), it resembles the cube in Figure 2.5(c).



(a) Numbered carbons in cubane.

(b) Cubane skeletal formula.



(c) A cube with vertices A, B, C, D, E, and F.

Figure 2.5: Cubane (C_8H_8) in its skeletal form resembles the cube polyhedron.

Can we utilize commonsense intuitions that are captured with polyhedra in the context of molecules? In the next section, we discuss existing general ontologies for shape and how they motivate the work that will be presented in the remainder of this dissertation.

2.1.3 General Ontologies for Shape with CardWorld & BoxWorld

In [Grü00], the author presents two ontologies for shape in the context of object recognition and computer vision: CardWorld[2] and BoxWorld[3], both of which are found online in the COmmon Logic Ontology Repository (COLORE). The *CardWorld ontology* is an ontology for two-dimensional (2D) object recognition in scenes with occlusion and images with noise, and has also been used in [GD09] as a basis for a cutting process ontology for sheet metal manufacturing. CardWorld represents two-dimensional objects by considering edges to be one-dimensional objects and the objects that contain edges are two-dimensional objects (called surfaces). Similarly, the *BoxWorld ontology* focuses on objects that have multiple connected surfaces in three dimensions (3D). With these two ontologies, shapes such as those seen in Figure 2.6 can be axiomatized in first-order logic.

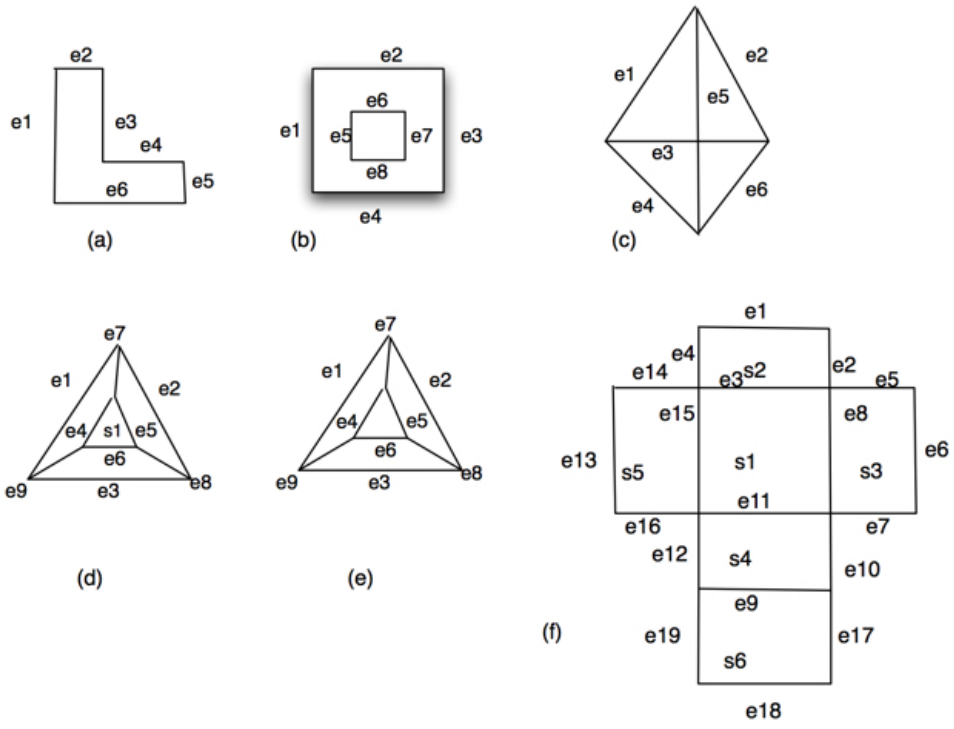


Figure 2.6: Shapes in CardWorld and BoxWorld. (Figure 1 in [GB11])

In addition to representing 2D and 3D shape, the axioms of CardWorld and BoxWorld also describe the relationships between edges and surfaces of boxes, and show that polyhedra are models of BoxWorld. Furthermore, CardWorld and BoxWorld have inspired other ontologies for physical objects, such as the PhysicalWorld Ontology that intends to capture physical phenomena (shape, location, connectedness, parthood, and kinetic and kinematic behaviour) [AG17].

[2] http://colore.oor.net/cardworld
[3] http://colore.oor.net/boxworld

As these are established ontologies for representing physical objects, we are interested in seeing how we can generalize notions of topological shape in the chemistry context. With the realization that polyhedral nets are 1-skeletons, we can leverage notions found in graph theory in our work. Consider the benzo[a]pyrene ($C_{20}H_{12}$) molecule in Figure 2.7: it is a molecule that contains five fused benzene rings together – we can see that these kinds of molecules are naturally occurring and resemble the polyhedral nets we discussed in Figure 2.4.

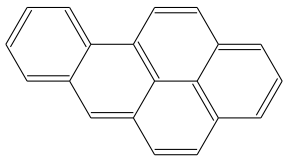


Figure 2.7: Benzo[a]pyrene ($C_{20}H_{12}$). This skeletal formula for this molecule resembles a polyhedral net.

Next, we discuss approaches taken by the chemistry and ontology communities to represent the topological shapes of molecules.

2.2 Conventional Approaches in Chemistry

Existing approaches to represent the shapes of molecules include chemical graph theory and cheminformatics approaches used to encode structural information in the form of identifiers. We discuss each of these in turn.

2.2.1 Chemical Graph Theory

Chemical graph theory describes chemical entities in terms of nodes and edges, which correspond to atoms and bonds, respectively [Tri92; Bal85]. Having its basis in mathematical graph theory, chemical graph theory has been widely adopted for denoting atomic composition and connectivity in chemical entities. Chemical graphs can be used to represent molecules, reactions, and clusters. The molecular graph formalism is used to describe atomic connectivity in a molecule by describing the nodes for atoms within the molecule, and the edges for the bonds between the atoms or groups; a shortcoming of chemical graph theory is that functional groups are not explicitly represented in the graphs [Bal85]; since it is imperative that substructures, such as rings and chains, need to be represented, chemical graph theory serves as a basis for comparison in terms of how to map the structural properties found in molecular structures with mathematical structures found in existing domain ontologies. We will go into more detail about the characteristics of chemical graphs in Chapter 5.1.

2.2.2 Cheminformatics: IUPAC, SMILES, & InChI

Chemical informatics ('cheminformatics') is the intersection of chemistry and information technology. It involves the use of computer and informational techniques, and plays a big role in the field of chemistry [Bro98]. In recent years, there has been an increasing interest in representing chemical entities in a computer-interpretable format to allow the determination and prediction of properties found in these chemical entities [Opr05]. Chemical structures have been the focus of more recent ontological work, as seen in [Deg+08; Has+11], that utilizes the Web Ontology Language (OWL)[4]. Motivated by the work done in [BM96; BM99], we are interested in *molecular structures* because these structures provide an opportunity to test the applications of ontologies in drug discovery; an ontology describing molecular structure will contain sufficient detail to carry out reasoning tasks that determine the properties of various molecular structures, either when combined or broken apart into substructures.

In practice, the primary application of cheminformatics is the storage, indexing, and searching of information related to chemical compounds. Tasks involved include information retrieval/extraction with unstructured data, and mining of structured data via databases, graphs, sequences, and digital libraries. Furthermore, virtual libraries of compounds can be generated to explore chemical space and to propose new compounds with desired properties. With respect to drug discovery, combinatorial chemistry utilizes virtual screening to computationally screen libraries of compounds to identify drug-like compounds that can dock with a given target; the focus of such screening tasks canvasses an enormous chemical space or a smaller library, depending on the desired properties. Furthermore, cheminformatics also involves the calculation of quantitative structure-activity relationship (QSAR) values to predict the activity of compounds and their structures using regression models.

With the advent of database systems in the 1960s, the earliest methods for handling chemical structure were based on chemical nomenclature and linear notation. The International Union of Pure and Applied Chemistry (IUPAC)[5] developed its own nomenclature system so that any chemical compound can be named with one set of standardized rules to avoid repeated names [Che06]. In addition to IUPAC nomenclature, mathematical graph theory can be used to model relations between connected objects and is known as the study of mathematical structures ('graphs'). Graph theory has found various applications in chemistry, especially in modelling molecular structures. The development of *chemical graph theory* has led to the development of ontologies that classify molecular structures in OWL, the most prominent being the Chemical Entities of Biological Interest (ChEBI).

In this section, we discuss each of these approaches in turn.

IUPAC Nomenclature The IUPAC nomenclature of organic and inorganic chemistry is a widely-accepted systematic method of naming organic chemical compounds. The purpose of this naming system is so that every possible organic compound has a name from which an unambiguous structural formula can be created [FDF74] – each chemical

[4] `https://www.w3.org/TR/owl-primer/`

[5] `http://www.iupac.org/`

name should refer to a single substance. IUPAC names convey information about the chemical make-up of a compound. As well, a limited number of alternative names for a substance is also acceptable but not the primary goal of chemical nomenclature.

To read IUPAC names of molecules, the longest continuous chain of carbon atoms is used to determine the basic root name of the compound. The presence of different functional groups that replace hydrogen or carbon atoms in the parent structure of a molecule is then used to modify the root name. The IUPAC naming convention can be divided into four parts: prefix, parent, locant, and suffix. The prefix identifies the various substituent groups in the molecule, the parent selects a main part of the molecule and indicates how many carbon atoms are in that part, the locants give positions of functional groups and substituents, and the suffix identifies the primary functional group (see Figure 2.8).

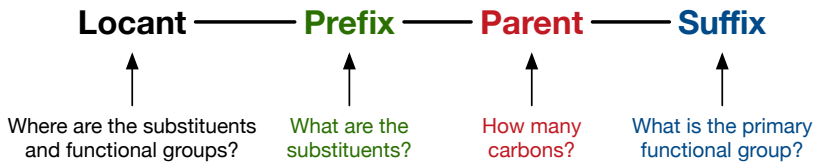
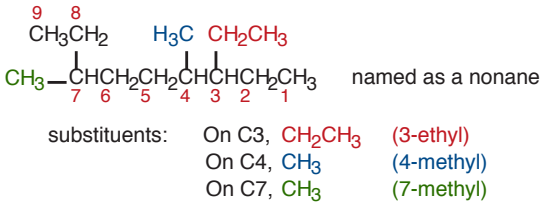


Figure 2.8: Components of an IUPAC name for a molecule.

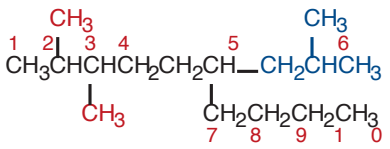
Examples of naming molecules are shown in Figure 2.9, where the coloured text indicates the corresponding component of the name. In each example, the longest continuous chain is outlined in black font, then the substituents and groups are identified in blue, green, and red font; carbons are also numbered in the examples to show the locants (numbers) of each substituent to locate its point of attachment to the parent chain. Hyphens are used to separate different prefixes and commas separate numbers. If two or more substituents are present, they are cited in alphabetical order in the resulting IUPAC name. We do not go over the nomenclature rules in detail here, as the rules are complex depending on the substituents and organization of functional groups, but we direct the reader to two official IUPAC publications, known as the Blue Book[6] and the Red Book[7], respectively; a condensed version of the IUPAC naming rules can also be found in Appendix A.

Official IUPAC naming recommendations are not always followed in practice – common or trivial versions of a compound's name may be used because the official IUPAC name is very long and is not as concise as the compound's structural formula. Most chemists communicate by means of structural diagrams, which are considered more important than the formal nomenclature rules [CGW12]. The IUPAC system has been criticized for applying extra naming rules to ensure that each possible compound has a unique name. This often leads to lengthy compound names that are not human-readable or foreign to most readers, which cause chemists to use the informal names of these compounds.

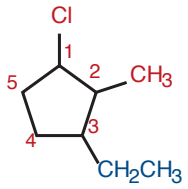
[6] http://pubs.rsc.org/en/content/ebook/9780854041824
[7] http://pubs.rsc.org/en/content/ebook/9780854044382



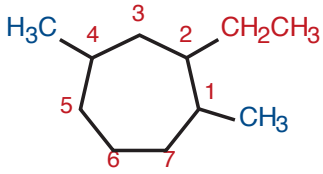
(a) 3-Ethyl-4,7-dimethylnonane ($C_{13}H_{28}$)



(b) 2,3-Dimethyl-6-(2-methylpropyl)decane ($C_{16}H_{34}$)



(c) 1-Chloro-3-ethyl-2-methylpentane (C_8H_{17})



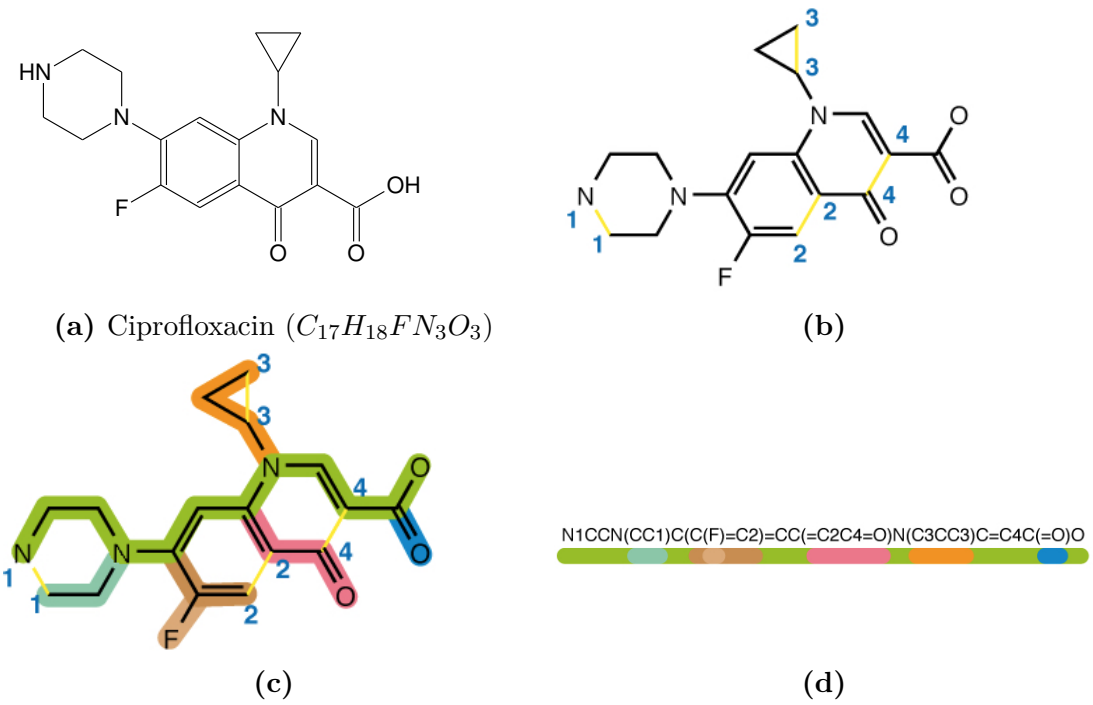
(d) 2-Ethyl-1,4-dimethylcyclohexane ($C_{10}H_{20}$)

Figure 2.9: Example of IUPAC naming for molecules.

Simplified molecular-input line-entry system (SMILES) A *proprietary* encoding format called the Simplified molecular-input line-entry system (SMILES) is heavily used within the chemistry community. Developed in the 1980s by David Weininger and Daylight Chemical Information Systems[8], SMILES is separate from IUPAC and is a line notation used to describe the structure of molecules using short American Standard Code for Information Interchange (ASCII) strings. SMILES is a widely used format in most molecule editors to convert the ASCII string into a two- or three-dimensional drawing of molecules. The strings are obtained by printing out the symbols for the elements found in the chemical graph, and follows a graph theoretic approach: hydrogen atoms and cycles are broken, then numeric suffix labels are used to indicate the nodes connected on the main carbon backbone, with parentheses used to indicate points of branching [Wei88]. This is graphically depicted in Figure 2.10, where the rings in ciprofloxacin ($C_{17}H_{18}FN_3O_3$, trade names: Ciloxan, Cpiro, Neofloxin) are broken in Figure 2.10(b) and are written as branches off the main backbone in Figure 2.10(c). Figure 2.10(d) outlines the SMILES string and the corresponding colours that were used in Figure 2.10(c).

As a *de facto* standard for the exchange of molecular structures, many software packages have been written in `C`, `C++`, `Java`, `Python`, and `LISP` to process SMILES strings. However, due to SMILES being proprietary, different chemical software developers have developed different SMILES generation and interpretation algorithms, which result in different SMILES versions for the same molecule; for example, Figure 2.11(b) outlines the various SMILES strings for the caffeine molecule. Consequently, SMILES strings obtained from different databases or research groups are not always interchangeable, unless these groups utilized the same software to generate or interpret the strings.

[8] `http://www.daylight.com/about/index.html`

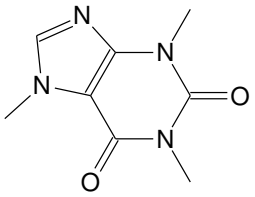


(a) Ciprofloxacin ($C_{17}H_{18}FN_3O_3$) **(b)**

(c) **(d)**

Figure 2.10: Generation of SMILES notation for ciprofloxacin ($C_{17}H_{18}FN_3O_3$). In (b), the rings in the molecule are broken first; then, in (c), the rings are written as branches off the main carbon backbone. The resulting SMILES string can be found in (d) as `c1c2c(cc(c1F)N3CCNCC3)n(cc(c2=O)C(=O)O)C4CC4`, where colours indicate the backbone and branches. (Images from [FD10])

Further, there also is a 'Canonical SMILES' that always outputs the atoms and bonds of any particular molecule in the exact same order, regardless of the source of the molecule or in software programs; a molecule's software, when expressed canonical SMILES, will always yield the same SMILES string. Canonical SMILES is only useful for a single database or for systems that were created using a single canonicalizer; algorithms used to generate canonical orderings of these SMILES strings go beyond the scope of this work, but we refer the reader to [WWW89] for more information. In Figure 2.11(b), the last string is the Canonical SMILES value for caffeine. Currently, the community is developing an open-standards version of SMILES called OpenSMILES[9] with a specifications document to promote universal adoption and revisions within the computational chemistry community [Jam16].



(a) Caffeine ($C_8H_{10}N_4O_2$)

1. `[c]1([n+]([CH3])[c]([c]2([c]([n+]1[CH3])[n][cH][n+]2[CH3]))[O-])[O-]`
2. `CN1C(=O)N(C)C(=O)C(N(C)C=N2)=C12`
3. `Cn1cnc2n(C)c(=O)n(C)c(=O)c12`
4. `Cn1cnc2c1c(=O)n(C)c(=O)n2C`
5. `O=C1C2=C(N=CN2C)N(C(=O)N1C)C`
6. `CN1C=NC2=C1C(=O)N(C(=O)N2C)C` (Canonical SMILES)

(b) Valid SMILES strings for caffeine.

InChI=`1S/C8H10N4O2/c1-10-4-9-6-5(10)7(13)12(3)8(14)11(6)2/h4H,1-3H3`
Key: `RYYVLZVUVIJVGH-UHFFFAOYSA-N`

(c) InChI value for caffeine.

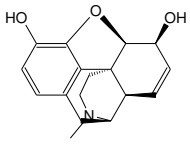
Figure 2.11: Various SMILES strings for a single molecule with a single InChI key.

IUPAC International Chemical Identifier (InChI) In 2005, IUPAC designed an open-source textual identifier for chemical substances known as the IUPAC International Chemical Identifier (InChI). The objective of this identifier is to provide a standard and human-readable way to encode molecular information and to ease the search of molecular information in databases and on the web. The InChI identifier describes molecules in terms of layers of information: the atoms and their bond connectivity, tautomeric information, isotope information, stereochemistry, and electronic charge information [Hel+15]. Unlike SMILES, InChI is open-source and is advertised as an identifier intended to link diverse data compilations and to promote unambiguous identification of chemical substances [InC18]. In addition to an identifier, there is also an InChIKey associated with each identifier: this is a 27-character signature based on the hash code of the InChI string. As we can see in Figure 2.11(c), the InChI for caffeine starts with a prefix either `InChI=1/` (any InChI) or `InChI=1S/` (Standard InChI), and an InChIKey string.

[9] `http://opensmiles.org/`

Like SMILES, molecule editors can generate different InChI strings for the same molecule, depending on the structural detail (e.g., tautomerism), and is also affected by the rules of (drawing) perception [InC18]. The Standard InChI was also launched in 2008 as version of InChI that maintains the same level of attention to structure details and conventions for drawing perception.

Issues with Nomenclature Despite IUPAC's intentions of developing a standardized nomenclature, we see the problems that arise from these quantitative intuitions of encoding molecular information: they are *not* human-readable, are intended for chemical software programs to *draw* molecules, and provide users with little insight on properties associated with the compound. For example, the chemical formula for morphine is $C_{17}H_{19}NO_3$ and is also referred to as the various names listed in Figure 2.12. Furthermore, there appears to be inconsistent approaches to adopting one particular encoding format since there are a multitude of options chemists can use (SMILES, canonical SMILES, InChI, Standard InChI, etc.). The semantics embedded in IUPAC and InChI identifiers are *implicit* in the data structures and the algorithms used to interpret them: IUPAC names involve string manipulations and InChI utilizes lists to unpack the structural information. Since the semantics for both are implicit, it inhibits sharing and reuse. Additional problems include the fact various equally valid SMILES strings can be written for a molecule, as illustrated in Figure 2.11 with the caffeine ($C_8H_{10}N_4O_2$) molecule. Additional problems, such as correctness and consistency, with IUPAC nomenclature, SMILES, and InChI are discussed in [OBo12] and [AKM12].



Chemical name [Nat15]:

```
Morphine; Morphinum; Morphia; Morphium; Morphin
```

IUPAC name:

```
(4R,4aR,7S,7aR,12bS)-3-methyl-2,4,4a,7,7a,13-
hexahydro-1H-4,12-methanobenzofuro[3,2-e]
isoquinoline-7,9-diol
```

SMILES:

```
CN1CC[C@]23C4=C5C=CC(O)=C4O[C@H]2[C@@H](O)C
=C[C@H]3[C@H]1C5
```

InChI identifier:

```
InChI=1S/C17H19NO3/c1-18-7-6-17-10-3-5-13(20)
16(17)21-15-12(19)4-2-9(14(15)17)8-11(10)18/h2
-5,10-11,13,16,19-20H,6-8H2,1H3/t10-,11+,13-,
16-,17-/m0/s1

InChI key: BQJCRHHNABKAKU-KBQPJGBKSA-N
```

Figure 2.12: Various ways of describing morphine ($C_{17}H_{19}NO_3$) using the chemical name, IUPAC name, SMILES identifier, and InChI identifier.

2.3 Existing Ontological Approaches

In this section, we go over existing ontological approaches to describing chemical structures. These include: the Chemical Entities of Biological Interest (ChEBI), the usage of Description Logic (DL) with chemical graphs, the usage of spatial reasoning, and an approach using monadic second-order logic.

Before we begin this discussion, we also discuss the needs for ontologies within the field of chemistry. In this section, we only present a subset of ontological approaches that pertain to small molecules and medicinal chemistry, but we also emphasize the fact that there has been a need for ontologies of molecular structure in work done by Dumontier et al. in [KDD08], [CD11], [DV09], and [VD07]. The need for knowledge representation in cheminformatics arises from the desire to integrate cheminformatics data from various sources – from our previous discussion of how structure is represented using IUPAC, SMILES, and InChI, we can see that the lack of standardization and the disparity across data formats encourages a need for common syntax and formal semantics to represent structure. The authors of [VD07] outline a simple OWL ontology of chemical functional groups to infer functional groups in organic compounds using the Pellet 1.4 reasoner in Protégé 4[10]. Due to the expressivity of OWL, ring structures are not identified with this ontology; instead, the authors illustrate how cyclic structures found in molecules can be identified using rules, written in the Semantic Web Rule Language (SWRL), to describe and reason about the structure and function of molecules. The approach taken in [VD07] is similar to the approach we present in Chapter 14.5 to identify functional groups in molecules; however, our approach has the added benefit that first-order logic can axiomatize cyclic structures to allow us to leverage graph theoretic properties of the resulting models, without the need for additional SWRL rules. While the primary foci of these papers are on Resource Description Framework (RDF) and OWL ontologies for chemical structure, the authors' comments and concerns still apply in our work with an ontology written in first-order logic.

2.3.1 Chemical Entities of Biological Interest (ChEBI)

The Chemical Entities of Biological Interest (ChEBI)[11] serves as a freely available dictionary of molecular entities for 'small' chemical compounds that are not macromolecules (biopolymers). This relational database contains information on groups and classes of chemical entities; the primary focus of ChEBI is to provide users with information on molecular entities that are natural or synthetic products found in living organisms [Deg+08]. The database is explicitly endorsed by IUPAC for general chemical nomenclature and Nomenclature Committee of the International Union of Biochemistry and Molecular Biology (NC-IUBMB) for biomedical nomenclature. As well, the ChEBI ontology can be exported into the OWL format to allow users to use it with tools such as Protégé, TopBraid, and the OWL-API [Has+13]. A screenshot of the ChEBI Ontology page for morphine[12] is shown in Figure 2.13, where the molecule name and identifier are

[10] https://protege.stanford.edu/

[11] http://www.ebi.ac.uk/chebi/

[12] https://www.ebi.ac.uk/chebi/chebiOntology.do?chebiId=CHEBI:17303

listed, along with the corresponding roles that the molecule has, as well as a listing of the child and derivative structures of the molecule.

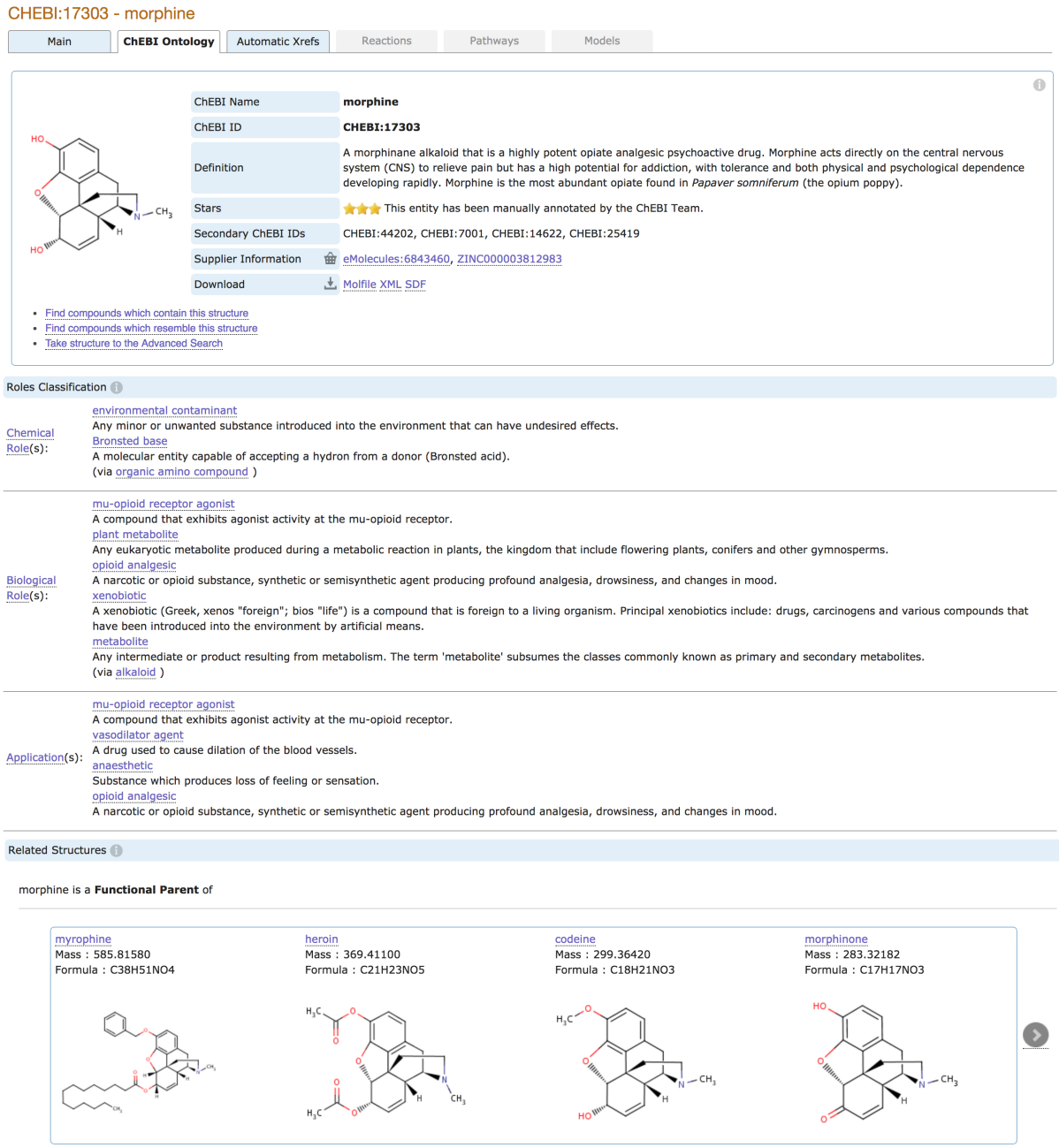


Figure 2.13: Screenshot of ChEBI Ontology page for morphine. The page lists the corresponding identifiers for the molecule in the ChEBI database, along with a list of roles and related structures.

We note that ChEBI lacks metadata information on the structural properties of the chemical compounds it describes. While ChEBI stores the two-dimensional (2D) and three-dimensional (3D) structural diagrams in the MDL Information Systems (MDL)

molfile format[13], it does not provide any semantic descriptors on whether a molecule is cyclic or tree-structured without having the user to view the graphical diagrams in an image viewer or through ChEBI's built-in applet for interactively viewing structural diagrams [Has+13].

2.3.2 Description Logic & Chemical Graphs

Similarly, the authors of [Has+10] analyze the ontological nature of chemical graphs and the relationships found between them in the context of ChEBI. Chemical graphs are interpreted as representations of mathematical connectivity objects to define known classes of molecules; the authors discuss which ChEBI relations range over classes and individuals and how these relations can be specified using Description Logic (DL), shown below in Figure 2.14. However, the authors' focus is primarily based on classification and reasoning among existing ChEBI roles and relations, and do not provide medicinal chemists with insight on the molecular structure.

Example IUPAC Derivation Relations

- Fluorobenzene-name $\equiv$ ($Name \sqcap \forall is\_about$.Flurobenzene)
- Fluorobenzene-name $\sqsubseteq \exists$is_derived_from.Benzene-name
- Benzene-CG $\sqsubseteq \exists is\_hydride\_graph\_of$.Flurobenzene-CG ... etc.

Example Class-to-Class Relations

- A is_a B $=_{def}$ $A \sqsubseteq B$
- A has_part B $=_{def}$ $\exists$has_part.B
- A has_parent_hydride B $=_{def}$ $A \sqsupseteq \forall has\_parent\_hydride B$ ... etc.

Figure 2.14: Example Description Logics for ChEBI from [Has+10]. The authors present IUPAC derivation relations, as well as class-to-class relations, in Description Logic. Please see [Has+10] for the full listing of rules.

Additionally, work done in [Mag13] and [MKH14] presents a DL approach to describe molecular structures. In particular, the focus is on representing cyclic molecular structures using nonmonotonic existential rules to ensure decidability of automated reasoning algorithms. In [Mag13], the author provides an in-depth overview of a rule-based framework that can be used to represent structured entities within a database, while work done in [MKH14] extends this by presenting a methodology that semi-automatically curates compounds in the ChEBI database by using this rule-based framework. Figure 2.15 shows examples of the types of DL rules to classify molecules.

While this work is interesting and further aids the curation of ChEBI content, we note that the expressivity of both OWL and DL limit the amount of molecular structure semantics that can be represented in these languages.

[13] A file format containing information about atoms, bonds, connectivity, and coordinates of a molecule.

Example DL Rules, TBox Axioms

r_1: $\bigwedge_{i=0}^{6} O(x_i) \wedge \bigwedge_{i=1}^{6} hasAtom(x_0, x_i) \wedge Molecule(x_0) \wedge \bigwedge_{i=1}^{6} bond(x_i, x_{(imod6)+1}) \rightarrow$ MoleculeWithSixMemberedRing(x_0)

r_2: $\bigwedge_{i=0}^{6} O(x_i) \wedge Molecule(x_o) \wedge \bigwedge_{i=1}^{6} hasAtom(x_o, x_i) \wedge \bigwedge_{i=1}^{6} Carbon(x_i) \wedge \bigwedge_{i=1,3,5} singleBond(x_i, x_{i+1}) \wedge \bigwedge_{i=2,4,6} doubleBond(x_i, x_{(imod6)+1}) \rightarrow$ MoleculeWithBenzeneRing(x_0)

α_4: Benzene $\sqsubseteq$ Molecule

α_5: Cyclobutane $\sqsubseteq$ Molecule

$\mathcal{R}_2 = \{singleBond \sqsubseteq singleBond^-, doubleBond \sqsubseteq doubleBond^-, singleBond \sqsubseteq bond, doubleBond \sqsubseteq bond\}$

Example Rule Translation from ChEBI Definitions

alkaneSuperClassOf saturated AND hydrocarbon AND NOT cyclic

$$saturated(x) \wedge hydroCarbon(x) \wedge \textbf{not}\ cyclic(x) \rightarrow alkane(x)$$

Figure 2.15: Example of DL rules and ABox in [Mag13] and rule translation from ChEBI definitions in [MKH14]. These rules determine whether a molecule is a six-membered ring, or whether a molecule contains a benzene ring. Please see [Mag13] for the full listing of rules.

Rules for Mercury Chloride ($HgCl_2$)

r_1: $mercuryChloride(x) \rightarrow \exists y_1 \exists y_2 \exists y_3. \bigwedge_{i=1}^{3} hasAtom(x, y_i) \wedge molecule(x) \wedge chlorine(y_1) \wedge mercury(y_2) \wedge chlorine(y_3) \wedge singleBond(y_1, y_2) \wedge singleBond(y_2, y_3)$

r_2: $singleBond(x_1, x_2) \rightarrow singleBond(x_2, x_1)$

r_3: $hasAtom(x_3, z_3) \wedge carbon(z_3) \rightarrow organic(x_3)$

r_4: $molecule(x_4) \wedge \textbf{not}\ organic(x_4) \rightarrow inorganic(x_4)$

r_5: $mercuryChloride(x_5) \rightarrow highlyToxic(x_5)$

r_5: $highlyToxic(x_6) \rightarrow \exists y_6. hasProperty(x_6, y_6) \wedge highToxicity(y_6)$

Figure 2.16: Example of a program of rules for mercury chloride in [Mag13, Example 1, pg. 16]. Rule r_1 describes the structure of mercury chloride in terms of its atoms and bonds. Rules r_2 and r_3 define classes of organic and inorganic molecules, and r_4 and r_5 pertain to toxicity.

2.3.3 Spatial Reasoning

In [Coh01], the author develops first-order axioms that describe spatial properties (mereology, topology, orientation, and distance) found in cell structure. In particular, the author axiomatizes DNA structure and provides axioms to distinguish between pairs, strands, and parts while simplifying hydrogen bonds found in DNA, example shown below in Figure 2.17.

We note that these axioms do not compose a general theory of shape (unlike CardWorld and BoxWorld discussed earlier): these axioms are equi-dimensional in nature and utilize spatial concepts found in mereotopology. In contrast to this work, what we would like to do is present a multidimensional approach to axiomatizing molecular shape, influenced by our discussion earlier with polyhedra and nets: we want to be able to write axioms for sets of atoms (vertices), bonds (edges), and functional groups (faces) *independent* of spatial relationships.

$$\forall x \forall y \, (Part(x,y) \supset \exists z \, (Part(z,y) \wedge (x \neq z))) \qquad \text{(CA-2)}$$

$$\begin{aligned} \forall x \, DnaChain(x) \supset (\forall n_1 ((Part(n_1,x) \wedge Nucleotide(n_1) \wedge \neg 5end(n_1)) \supset \\ \exists! n_2 (Part(n_2,x) \wedge Nucleotide(n_2) \wedge \\ Interlocked(n_1,n_2)))) \wedge \forall n_2 \, ((Part(n_2,x) \wedge \\ Nucleotide(n_2) \wedge \neg 3end(n_2)) \supset \\ \exists! n_1 \, (Part(n_1,x) \wedge Nucleotide(n_1) \wedge \\ Interlocked(n_1,n_2)))) \end{aligned} \qquad \text{(CA-8)}$$

$$\begin{aligned} NitrogenRingCompound(x) \equiv_{df} \exists r \, (Part(r,x) \wedge Ring(r) \wedge \\ \exists p \, (Part(p,r) \wedge Nitrogen(p))) \end{aligned} \qquad \text{(CD-6)}$$

$$Fused(x,y) \equiv_{df} \exists z \, (Atom(z) \wedge Part(z,x) \wedge Part(z,y)) \qquad \text{(CD-16)}$$

$$\begin{aligned} HBond(z_1,z_2) \equiv (Pyrimidine(z_1) \wedge Purine(z_2)) \vee \\ (Pyrimidine(z_2) \wedge Purine(z_1)) \end{aligned} \qquad \text{(CT-2)}$$

Figure 2.17: Example axioms from [Coh01] for biospatial knowledge. The identifiers 'CA', 'CD' and 'CT' denote axiom, definition, and theorem, respectively, and numbers correspond to the identifiers used in [Coh01].

While this work provides us with a first-order axioms for *spatial* representation in biology, its reliance on mereotopology imports ontological commitments which are irrelevant for the shape concepts we are interested in; for example, we are not concerned with how nucleotides are positioned in DNA chains. However, we acknowledge that this work provides us with a point of reference should we need to re/use some of the mereotopological notions in future work with molecular reactions.

2.3.4 Monadic Second-Order Logic

The authors of [HKM11] present an alternative framework to model the shapes of molecules with the combination of monadic second-order reasoning over chemical graphs and topo-

logical features of chemical molecules. Their primary motivation for this framework is due to the development of highly symmetrical, polycyclic chemical entities that are made up of mainly carbon atoms, such as buckminsterfullerene molecules (bucky-balls, C_{60}), nanotubes, nanostrips, and molecular knots. These molecules are primarily characterized by their shape or topology, rather than their parts or names. The authors use Monadic Second-Order Logic (MSOL) to express the graph properties and to define graph classes for such molecules, which allows them to verify whether a chemical graph (presented in a MDL molfile) falls into a graph class defined by a MSOL theory, shown in Figure 2.18. As well, they utilize a hyperontology framework that harnesses various heterogeneous logics and formalisms to further refine ontologies formalized in different logics.

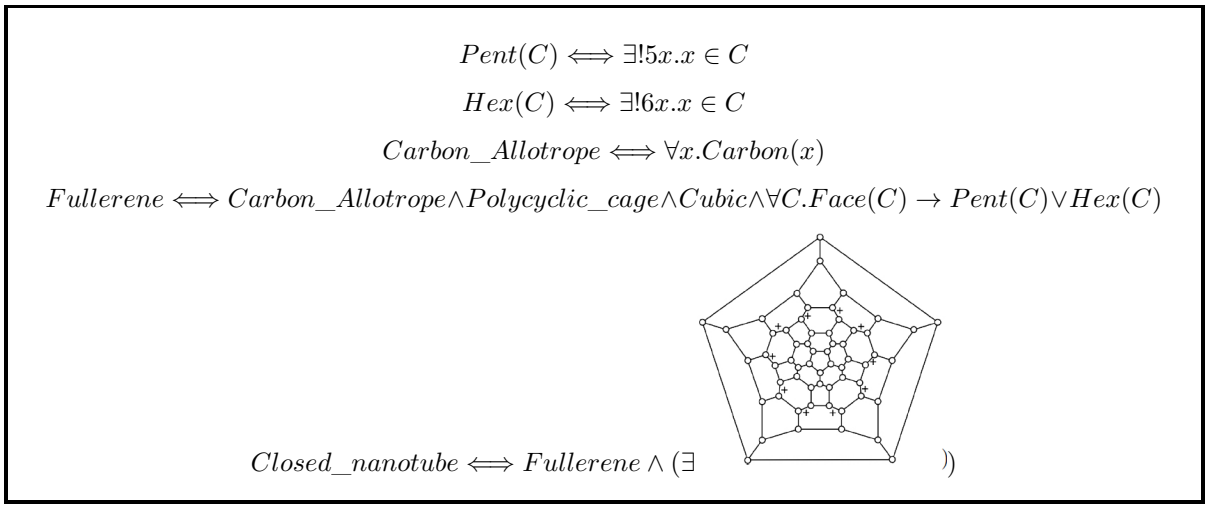


Figure 2.18: Example axioms from [HKM11] for graph classes using MSOL.

Why use first-order logic? In the axiomatization of [Mag13], bonds between atoms are represented as relations, such as $bond(x, y)$ and $singleBond(x, y)$ in Figure 2.15. A similar approach is taken in [HKM11], where the $edge(x, y)$ relation is used to describe bonds in the graph axioms seen in Figure 2.18. In both approaches, the authors represent bonds as relations in the domain to axiomatize these bonds as *sets of pairs of atoms*. Since the approach used in [HKM11] deals with sets of pairs of atoms, it makes sense that the authors utilized second-order logic to quantify over them.

While [HKM11] provides us with a methodology of how to apply second-order logic to model molecular structure, we wish to only use first-order logic to demonstrate the reasoning capabilities of the molecular structure ontology developed in this project. Second-order logic is incomplete, whereas first-order logic is complete [Göd92], and that there are established reasoners for the latter, not the former. As we will see in Parts II and III, using first-order logic to axiomatize our molecular structure ontology will let us reuse other first-order ontologies with established first-order reasoners and model-builders, such as Prover9-Mace4[14] or Vampire[15].

[14] `https://www.cs.unm.edu/~mccune/mace4/`
[15] `https://vprover.github.io/`

With an *impoverished* ontology like the one in [HKM11], second-order logic is required to axiomatize and refer to the bonds since the sets of atoms are quantified in the domain[16]. Since bonds are relations in [HKM11], second-order logic is required to refer to the bonds as elements of the domain (these pairs of atoms are then referred to as sets in the domain). In contrast, by having a *richer* ontology where bonds are classes in the domain, there is no need to utilize second-order logic to axiomatize bonds. Furthermore, we want to be able to axiomatize rings as classes in the domain as well; using the approach in [HKM11], rings and chains found in molecules would need to be represented as sets of atoms *and* bonds, which requires second-order logic. In our axiomatization that we discuss in Part I, we adopt a multidimensional approach by having atoms, bonds, and groups as elements of the domain in the first-order.

2.4 Shortcomings of Existing Approaches

We have reviewed the conventional and ontological approaches to describing molecules, and found that these efforts fall short of representing the structure of molecules. Ternary predicates cannot be represented in OWL unless they are reified, which may pose a problem when modelling properties in the chemical domain. Since OWL is not expressive enough for the reasoning tasks we have in mind and IUPAC nomenclature is not human-readable enough for a user to decipher the structural properties of a molecule, we have decided to focus on utilizing first-order logic to develop an ontology that fully captures the structural semantics of molecules named by IUPAC. We have seen that the primary focus of the community is to develop OWL ontologies that correctly capture the hierarchical relationships between molecules, but not their structural properties. Likewise, chemical graph theory and IUPAC do not capture the structural properties for molecules that we wish to axiomatize in an ontology. We summarize these *shortcomings* of existing work as follows:

1. Conventional approaches using chemical graph theory are able to represent the structure of a molecule; however, the inability to represent functional groups in chemical graph theory hinders its ability to be used for reasoning about structural properties. As well, reasoning with these graphs is conducted through the usage of special algorithms. IUPAC nomenclature tells a chemist how to *draw* the underlying molecular graph but does not clearly identify the *topological* structure of the molecule: rings are not reified in IUPAC; i.e., the term '*tricyclo*' does not clearly identify *which* components of the graphs are the three rings in the structure.

2. Current ontological approaches primarily focus on the classification of molecules, provide few semantics about chemical structure that can be used for reasoning, and do not properly represent structure and shape of the molecule. By '*structure*', we mean the decomposition of the molecule into its parts (rings and chains) and the relationships found between these components. In addition to IUPAC nomenclature, current ontological approaches do not clearly capture these components and relationships.

[16]For a quick overview of second-order logic, see Appendix B.5.

Intended uses for the ontology would be to primarily to allow users to *reason* about molecular structure, in addition to providing a more elegant and human-readable format for axiomatizing chemical compounds and their substructures. As well, a molecular structure ontology will serve as an ontology of 'building blocks' – users can build up big molecules by picking and choosing components of the ontology to combine. Since the process of synthesizing new molecules is tedious and meticulous for chemists, the development of this ontology can allow them to combine functional groups and atoms together in an ontological fashion to determine the types of reactions, structural properties, or molecular compounds that will result from the combinations. Current work done in the chemistry community indicates that there is an interest in the automatic synthesis of molecular building blocks of simple drugs [Ser15]; with an ontology, it would be possible to develop software systems to provide decision support for drug discovery.

Chapter 3

Requirements for a Molecular Structure Ontology[1]

With a molecular structure ontology, the structural properties of molecules that are named by IUPAC nomenclature can be represented in logic. By using first-order logic, it is possible to represent properties and semantics that are not captured by IUPAC, ChEBI, and chemical graph theory. This means that, for any structural formula that exists, we can axiomatize these graphical representations using the ontology and can distinguish between the bond types (single, double, and triple), atom types (C, N, O etc.), and whether the compound contains rings or chains.

3.1 Usage of First Order and Common Logic Representation

In order to capture the semantics of concepts utilized in this work, first-order logic[2] is utilized due its expressive power and usages in the ontologies we have examined.

3.1.1 Common Logic (CL)

We utilize a repository environment to store these computer-interpretable ontologies and theories in the Common Logic (CL) syntax[3]. Common Logic is a standardized logical language for the specification of first-order ontologies and knowledge bases, and its details can be found in the ISO 24707:2007 document [Int07]. The author of [Has08] discusses the flexibility of the Common Logic Interchange Format (CLIF) and its ability to support the high-level of expressibility in first-order logic. Consequently, all of the ontologies and theories found in the repository environment are written in Common Logic. We are able to examine meta-theoretic relationships between ontologies found in the COmmon Logic Ontology Repository (COLORE) and, where applicable, prove them based on the first-order notions of interpretability and representation. The soundness and completeness of

[1] This chapter extends the work previously published as [CG16].

[2] For a quick overview of first-order logic, see Appendix B.1.

[3] For a quick overview of the Common Logic syntax, see Appendix B.3.

first-order logic aids us in the verification of theories: anything proven using the axioms of a theory holds for all possible models of that theory. For readability purposes, axioms are written in the traditional first-order logic syntax in this work, and the CLIF versions can be found online in the repository, indicated in a footnote with the corresponding hyperlink to its location in the repository.

3.1.2 The COmmon Logic Ontology REpository (COLORE)

The COmmon Logic Ontology Repository (COLORE)[4] is an open repository of first-order ontologies that serves as a test environment for the design, evaluation, and application of these ontologies. The existence of the repository gives the community a common foundation for developing complex ontologies and allows the exploration and examination of stored ontologies in an efficient and directed manner. Ontologies that share a similar domain are explicitly linked in the CLIF files, allowing users to explore a *hierarchy* composed of the related ontology modules, along with any extensions derived from mapping modules together. All theories within the repository are organized into hierarchies; a detailed discussion of the organization of theories within hierarchies can be found in [Grü+12].

Since each module of an ontology represents a different set of ontological commitments, having the repository connect all ontologies that share logical similarities allows greater reuse of these modules; for example, when two ontologies are connected through the repository, they are able to use translation definitions within the repository to share their modules. Thus, as the repository grows, the number of semantic integration possibilities increases as well to enable users to gain a better understanding of what information can be shared between modules along with the various relationships between these modules. First-order theories, the notions of hierarchies, conservative extension and non-conservative extension are discussed in detail in Chapter 11.

3.1.3 Verification of Ontologies

With respect to 'checking' whether the ontology accurately represents the notions of structure and properties of molecules, we verify it with a set of theories found in geometry. To verify an ontology, we apply model-theoretic notions to characterize the semantics of an ontology as a set of *intended structures*[5]. We specify these structures with well-understood mathematical theories to determine whether the axiomatization of an ontology matches its intended models; these theories include partial orderings, lattices, incidence structures, geometries, and algebra [Grü11; Grü+10]. If an ontology's axiomatization contains *unintended* models, then it is possible to find sentences that are entailed by the intended models, but these sentences are not provable from the axioms of the ontology. Such models provide barriers to semantic interoperability between software systems and may prevent the entailment of sentences [Grü11; Grü+10].

[4] COLORE can be accessed via `http://colore.oor.net`

[5] Adopted from [Grü11]: for the ontology, an intended structure is a set of structures that characterizes the semantics of an ontology's terminology.

By verifying an ontology, we would like to characterize its models up to isomorphism to determine whether these models are *equivalent* to the intended structures of the ontology [Grü11]. To do this, we utilize the mathematical notion of *representation theorems*, where we prove that every intended structure is a model of the ontology and that every model of the ontology is *elementary equivalent* to some intended structure. We leverage work done on theories for incidence structures to verify the ontology presented in this work, and discuss this further in Chapters 12 and 13.

3.2 Determining Requirements via Competency Questions

To guide the design of the ontology, we used competency questions to determine the scope of the ontology: they are essentially questions that a knowledge base, based on the ontology, should answer. These questions aid ontology designers in determining whether the ontology contains information to answer these types of questions, and whether a particular level of detail or representation is needed for the ontology. Below are three categories of high-level competency questions the ontology should answer.

3.2.1 Similarity-Based Competency Questions

Competency questions in this category pertain to the shape and structure of molecules. Examples of competency questions include the following:

(**CQ-1**) Which molecules have common substructures with penicillin?

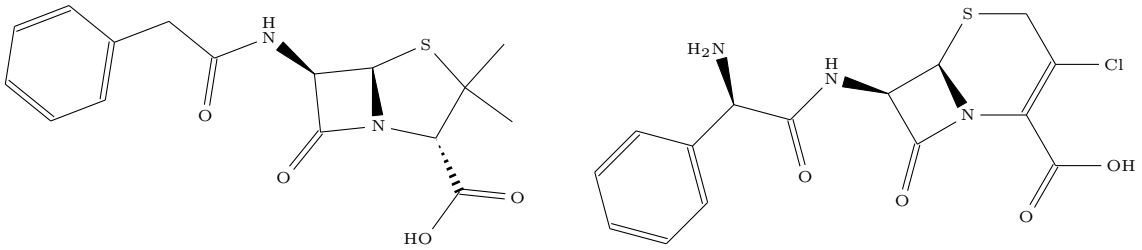
(**CQ-2**) Which antibiotics contain a β-lactam ring?
For example, Figure 3.1 illustrates how benzylpenicillin (commonly known as 'penicillin') and cefaclor are both similar in structure, where the four-membered β-lactam ring is fused to a five-membered thiazolidine ring and a six-membered ring containing a sulphur, respectively. Queries pertaining to these molecules include determining how they are different, such as the phenylacetatimide and phenylglycinamide attached to β-lactam rings in both.

(**CQ-3**) What are molecules that contain two fused rings?

(**CQ-4**) Which molecules contain a given functional group?

3.2.2 Substitution

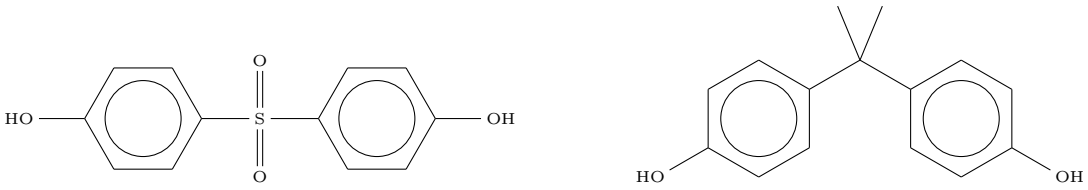
Competency questions in this category pertain to substitutions of elements and/or sub-molecules that can be made when molecules are of the same shape. An example of a competency question is the following:



(a) Penicillin ($C_9H_{11}N_2O_4S$) **(b)** Cefaclor ($C_{15}H_{14}ClN_3O_4S$)

Figure 3.1: Structural similarities between penicillin and cefaclor. Both molecules contain a β-lactam ring that is fused to different components in both.

(**CQ-5**) What molecules are equivalent to molecule x after we substitute substructure y with substructure z?
For example, Figure 3.2 illustrates how Bisphenol S (BPS) and Bisphenol A (BPA) are both similar in structure, where the dimethylmethylene group ($C(CH_3)_2$) in BPA is replaced with a sulfone group (SO_2) in BPS.



(a) Bisphenol S (BPS) **(b)** Bisphenol A (BPA)

Figure 3.2: Structural similarities between Bisphenol A and Bisphenol S. Both molecules retain the same shape with different components substituted in both.

3.2.3 Synthesis

Competency questions in this category pertain to the composition of the building blocks of molecules: atoms, bond type, and functional groups. An example of a competency question is the following:

(**CQ-6**) What molecules can be composed to form a new molecule with given structural properties?

3.3 Requirements for the Equivalence for Models and Molecules

Now that we have the competency questions to help guide the design of the ontology, there is also the issue on how models of the ontology may correspond with molecules in reality. How do we distinguish between chemically-feasible and chemically-infeasible molecules with the ontology? How do we know whether our conceptualization of molecular structure corresponds to the real-world notions of molecules?

We briefly discuss the notions of conceptualization in [GOS09]. In this seminal work, the authors discuss how perception and conceptualization affect the development of an ontology and its resulting models. The authors outline the following in [GOS09]:

> "C is a conceptualization, L is a logical language with vocabulary V, and K is an ontological commitment. An ontology O_K for C with vocabulary V and ontological commitment K is a logical theory consisting of a set of formulae L, designed so that the set of its models approximates as well as possible the set of intended models of L according to K."

This is graphically presented in Figure 3.3, where we intend for the proposed ontology to be as best an approximation to representing molecular structure. Simply put, an axiomatization is semantically correct *if and only if* it does not include any unintended models and does not omit any intended models [Kat16].

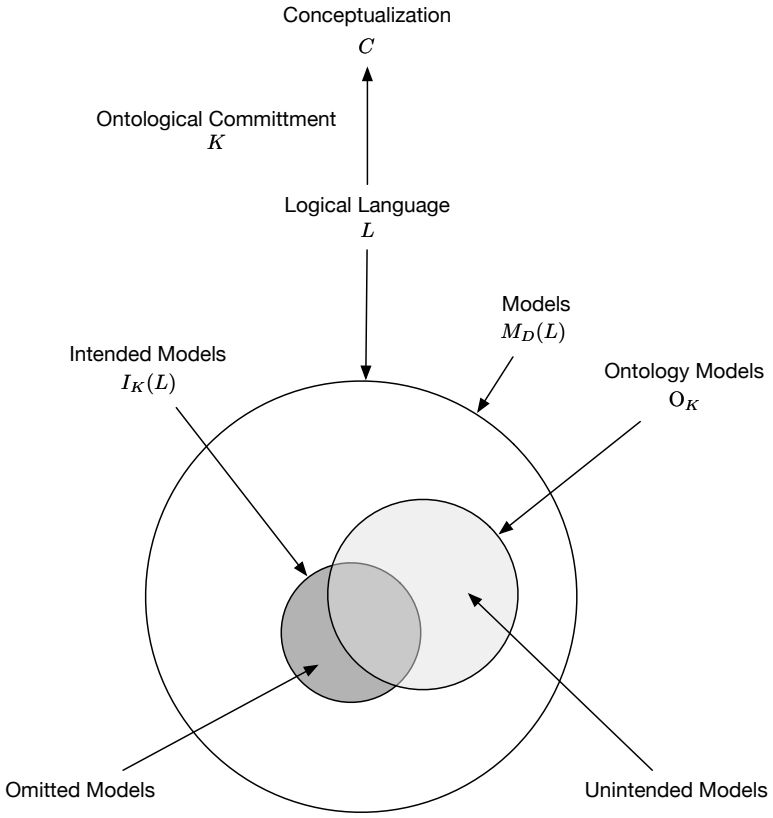


Figure 3.3: Relationships between the intended, unintended, and omitted models of an ontology and its conceptualization. (Figure 2 in [GOS09])

While competency questions capture the structure requirements of molecules, there is an issue of whether there is a mismatch of models of the ontology with (potential) molecules. With the ontology, can we capture a chemist's knowledge about molecular structure and describe it accordingly? We revisit this issue in greater detail in Chapter 14, which discusses how the ontology is used as a technique for model construction, but this raises the question of what exactly is the correspondence between models generated by

the ontology and what currently exists in chemistry, and representing heuristics to help chemists discover molecules; for example, a query with the ontology can be: "construct molecules that has this particular property." If medicinal chemists have hunches and heuristics to follow, we are able to represent the property with the ontology: these properties can then be used as constraints that need to be satisfied when building a first-order model of the ontology.

As a result, we can add the following requirements for the equivalence of models and molecules with the proposed ontology:

(**MOD-1**) There should be a 1-to-1 correspondence of models of the ontology and molecules in chemistry. To ensure models of the ontology are accurate, we will utilize graph theory and incidence structures found in mathematics. We discuss the details of utilizing graph theory in Part I and incidence structures in Part II.

(**MOD-2**) **Intended models** of the ontology are molecules. Relating this back to Figure 3.3, all intended models of the ontology correspond to a chemically-feasible molecule.

(**MOD-3**) **Unintended models** of the ontology are not molecules; these may be chemically infeasible molecules that cannot be physically realized or actually are impossible to synthesize.

(**MOD-4**) **Omitted models** are molecules that are not models of the ontology. In Chapter 14, we will discuss how there do not exist any omitted models in the ontology due to the attachment theorem used to compose parts of molecules together.

With the ontology, we need to ensure that we have an appropriate set of axioms to prevent unintended models and to prevent omitted models. For example, if we take out an axiom in the ontology, we may get unintended models such as molecules that are only rings that are only fused together – this is not reflective of existing molecules since some might not consist only of rings, or rings that are not fused together but bonded together in different ways. Omitted models might occur if we include or exclude axioms that prevent them from resembling molecules. We discuss these design decisions of what to include and exclude in detail in Part I.

3.4 Ontological Commitments as Requirements

Based on the competency questions, we can extract requirements that drive the design of the ontology to ensure that we capture all of the information required to represent chemical structure. Given that the eventual application of our molecular structure ontology is to be used with drug discovery, we identified the following:

(**R-1**) The ontology must represent the properties of elements, functional groups, connections between functional groups and components of molecules, along with a classification of molecules with respect to their structures.

(**R-2**) The ontology must represent molecules as graphs, such that molecules can be decomposed into their primitive functional groups.

Ontological commitments are intuitions about a particular set of concepts that are captured by an ontology's axioms. We have identified the following *ontological commitments* that the axioms need to capture about chemical structure:

(**R-3**) Atoms, bonds, and functional groups are elements of the domain (*things* in the ontology).

(**R-4**) Primitive functional groups are *rings* and *chains*, which correspond to induced cyclic and path subgraphs, respectively.

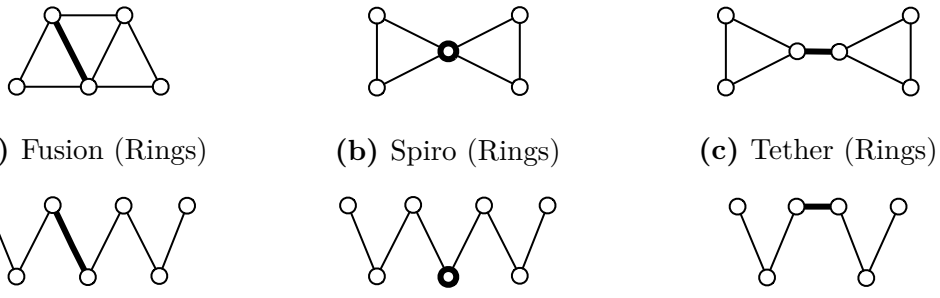
(**R-5**) The carbon backbone of organic molecules forms the essential structure of a given compound, which we call a '*skeleton*'. The skeleton consists of the various combinations of rings, chains, and atoms that are not in any functional groups.

Functional groups can be categorized according to how groups can share atoms or are bonded together (refer to Figure 3.4); we categorize the various types of connections as requirements for the ontology:

(**R-6**) The ontology must represent the *fusion* connection: there is an overlapping bond between the groups – any two rings or chains share a bond (see Figures 3.4(a) and 3.4(d)).

(**R-7**) The ontology must represent the *spiro* connection: rings or chains share an atom: they overlap at one atom (see Figures 3.4(b) and 3.4(e)).

(**R-8**) The ontology must represent the *tether* connection: no atoms between the groups are shared, but any two groups are bonded together by a bond between each group (see Figures 3.4(c) and 3.4(f)).



(a) Fusion (Rings) **(b)** Spiro (Rings) **(c)** Tether (Rings)

(d) Fusion (Chains) **(e)** Spiro (Chains) **(f)** Tether (Chains)

Figure 3.4: Types of connections between functional groups (rings and chains). The thicker circles and lines in the subfigures indicate the atoms and bonds by which the functional groups are connected.

3.5 Reasoning Problems & Intended Applications

The types of reasoning problems and intended applications for the ontology include the following:

1. *Information retrieval*: this pertains to using the ontology to deduce facts (entailment). For example, this would involve developing queries to answer additional competency questions chemists might have about molecules:

   (**CQ-7**) "Find all molecules that are only fused together."

   (**CQ-8**) "What are molecules that do not have fused rings?"

2. *Design of molecules*: this pertains to using the ontology to answer satisfiability problems (model construction). We are interested in determining whether axioms of the ontology generate models that correspond to molecules (and vice versa). For example, this pertains to constructing a model of the ontology that satisfies whatever query, complete or not, the chemists input:

   (**CQ-9**) "Construct a molecule where no two rings are fused together."

   As well, our interest in satisfiability problems lies in discovering techniques for model construction using the ontology.

As we will see in Chapter 14, the proposed ontology is meant to assist medicinal chemists with both their search for molecules with certain properties and to help them design *new* molecules through the decomposition and composition of primitive functional groups.

Part I

The Molecular Structure Ontology (MoSt)

Chapter 4

Overview

In this part of the thesis[1], we introduce the Molecular Structure Ontology (MoSt) which adheres to the requirements we have identified in Chapter 3.4. The ontology has been axiomatized in the Common Logic syntax and is available in the $\mathbb{H}^{most}$ hierarchy in COLORE[2]. The relationships among the theories in MoSt are depicted informally in Figure 4.1, where arrows indicate relationships between theories. The MoSt theories import the molecular graph axioms (grouped as 'most_graph' in the $\mathbb{H}^{molecular\_graph}$ hierarchy[3]). We will discuss the relationships between the theories in detail in Part II – for now, we present this simplified hierarchy diagram to illustrate the semantic conditions that inspired the organization of the MoSt axioms, and also use this hierarchy to illustrate how the axioms will be presented in this part of the dissertation.

Before we begin to outline all of the ontological choices that were made, we will go over the methodology used to develop the ontology. This is particularly important, as there are some ontology design decisions made that do not follow the IUPAC nomenclature. The approach taken can be summarized, by chapter, as follows:

Chapter 5: All molecules consist of atoms and bonds. Chemical graph theory examines these relationships between atoms and bonds. How can we axiomatize these notions?

Chapter 6: Now that we have an underlying chemical graph that represents the atoms and bonds, it does not make sense to only focus on the atoms and bonds of a molecule. What about the (topological) shape of a molecule? How do we identify whether atoms bonded together resemble a 'ring' or a 'chain'?

Unlike previous work done in [Tri92], [Coh01], [Mag13], and [MKH14], we want to be able to look at molecules from the perspective of their 'building blocks': these are the *functional groups* found in chemistry.

Chapter 7: While axiomatizing the functional groups, we noticed that we can leverage notions found in graph theory to aid with the axiomatization process: all graphs have cycles and paths, which can correspond to rings and chains

[1] This part of the thesis extends the work previously published as [CG16].

[2] `http://colore.oor.net/most/`

[3] `http://colore.oor.net/molecular_graph/`

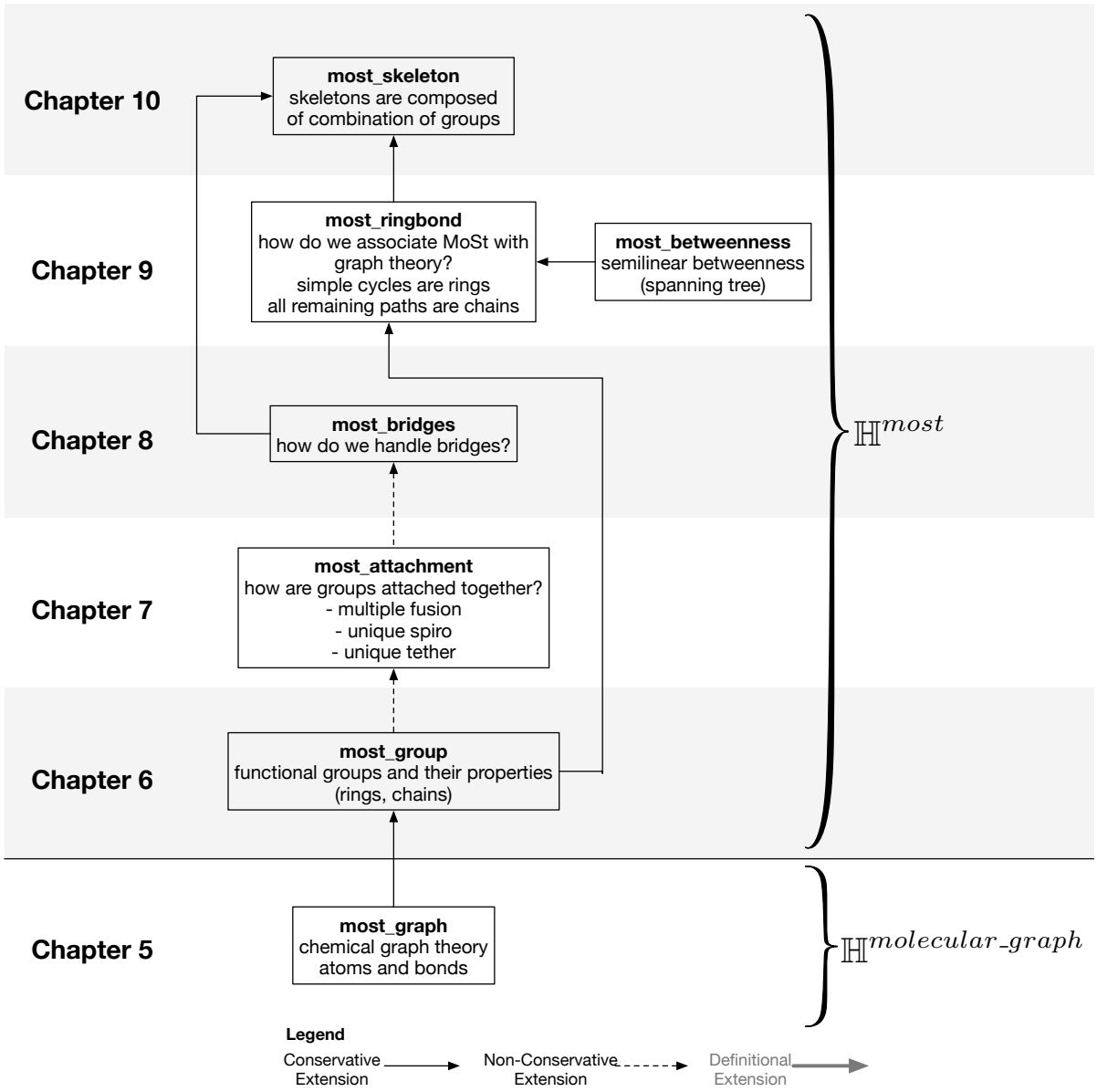


Figure 4.1: Organization of theories in the $\mathbb{H}^{most}$ and $\mathbb{H}^{molecular\_graph}$ hierarchies in COLORE; the names of the theories listed are the Common Logic Interchange Format (CLIF) files that correspond to the theories in the hierarchies. Solid arrows indicate conservative extension, dotted arrows indicate non-conservative extension, and bolded grey arrows indicate definitional extension. Brace brackets indicate the organization of these theories into hierarchies in COLORE.

found in molecules, respectively. We can therefore utilize existing mathematical notions in geometry in our first-order representation to account for functional groups, molecules, and skeletons as elements of the domain. In our observations from drawing out molecular graphs, we made note of similarities between molecules and shapes that only consist of rings (polycyclic molecules). We describe the properties of these molecules and identify axioms that can be used to describe such features.

Further, we examine case studies of molecules with various attachments. We first look at polycyclic molecules, which are purely fused molecules, and discuss how to axiomatize this restricted class of molecules. Since not all molecules are fused, nor are they all rings, how can we relax constraints in the axioms to allow chains to be part of molecules?

If we continue to relax constraints, how can we introduce ways of combining chains and rings together? Recall Requirements (**R-6**), (**R-7**), and (**R-8**), how do we handle these attachment requirements with respect to what we have axiomatized so far?

Chapter 8: There are molecules that are considered to have 'bridges' in chemistry – how can we represent these with the axioms we have? Do we require more axioms in the ontology to handle bridges?

Chapter 9: Now that we have a basis for axiomatizing various molecules and their attachments, what can we leverage from graph theory? In our discussion on how to axiomatize the various attachments between functional groups, we also noticed that some molecules have cycles in the underlying graph that do not necessarily correspond to chemical rings. What approaches can we take from graph theory to ensure these cycles are actually chemical rings?

Chapter 10: Finally, we discuss how skeletons play a role in the overall design of the ontology, as they pull all of the axioms presented in the previous chapters together.

To ensure the reader's full comprehension of this work, regardless of their background in chemistry or ontologies, we have structured this part of the thesis to present the ontology in a pedagogical fashion. Chemical concepts are first presented and discussed, along with any relevant examples, and are then supplemented with the first-order axiomatizations of these concepts.

Chapter 5

Chemical Graph Theory: An Atomic Approach

In this chapter, we provide an overview of the existing chemical graph theory approach to describe molecules as two-dimensional graphs. Prior to presenting our first-order axiomatization of the concepts in this chapter, we will also briefly go over some basic concepts in molecular graph theory and chemistry.

5.1 Characteristics of Molecular Graph Theory

In chemical graph theory, a molecular graph or chemical graph is a representation of the structural formula of molecule; it is represented in the terminology of graph theory found in mathematics. We utilize the definition of graph found in [Die12], which defines a graph as sets of pairs of vertices and edges in Definition 5.1.

Definition 5.1: Graph (Adopted from [Die12])
A graph is a pair $G = (V, E)$ of sets such that $E \subseteq [V]^2$; thus, the elements of E are 2-element subsets of V. The elements of V are the vertices (or nodes, or points) of the graph G, the elements of E are its edges (or lines). See Figure 5.1.

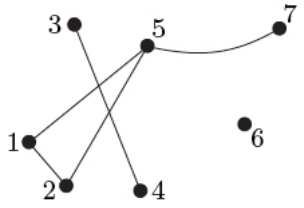


Figure 5.1: The graph on $V = \{1, 2, 3, 4, 5, 6, 7\}$ with edge set $E = \{\{1, 2\}, \{1, 5\}, \{2, 5\}, \{3, 4\}, \{5, 7\}\}$. (Figure 1.1.1 in [Die12]).

Chemists have leveraged this mathematical structure of graphs by introducing the notion of a *chemical (or molecular) graph*[1]. As defined in Definition 5.2, a chemical

[1]In this thesis, we use the terms 'chemical graph' and 'molecular graph' interchangeably.

graph is a *labeled graph* – atoms of the compound correspond to the vertices of the graph, and the chemical bonds of the molecule correspond to the edges of the graph. Vertices are labelled with the the corresponding atoms and edges are labelled with the types of bonds (for example, single or double bonds).

Definition 5.2: Molecular Graph (Adopted from [Bon91])
A molecular graph $G = (V, E)$ is a graph that has $n = |V|$ nodes and $m = |E|$ edges. The nodes $v_i \in V$ represent non-hydrogen atoms and the edges $(v_i, v_j) \in E$ represent covalent bonds between the corresponding atoms. See Figure 5.2.

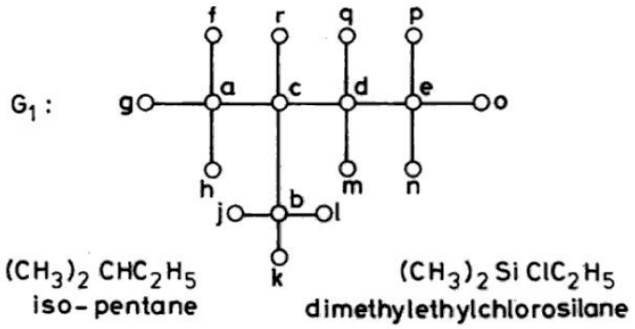


Figure 5.2: The iso-topological molecular graph for isopentane, $(CH_3)_2CHC_2H_5$, and dimethylethylchlorosilane, $(CH_3)_2SiClC_2H_5$, respectively. (Graph G_1 in [Bon91])

Following the seminal work on chemical graphs in [Tri92], we provide a mapping of how graph-theoretical terms correspond to concepts found in chemistry in Table 5.1. As chemistry is carried out by means of drawing diagrams to graphically depict compounds, graph theory provides an avenue of providing chemists with qualitative predictions about the structure and reactivity of compounds [Tri92].

Table 5.1: Correspondences between graph-theoretical and chemistry terminology.

Graph-Theoretical Term	Chemistry Term
Vertex (point)	Atom
Edge (line)	Chemical Bond
Path	Chain (e.g., linear alkane or polyene)
Cycle	Ring (e.g., cycloalkane or annulene)

However, we make the distinction from the work presented in [Tri92] with our interpretation of a chemical graph: we *allow* multiple edges in the graph, whereas the graphs presented in [Tri92] are "finite, undirected graphs without multiple edges and loops," thereby meaning these chemical graphs can only represent single bonds. The reason for our distinction is due to the fact that we utilize first-order logic to describe the graphs and allow for chemical graphs to be represented as multigraphs. In mathematics and

graph theory, multigraphs are graphs that are permitted to have multiple edges between the same node pair (i.e., edges that have the same end nodes), which allows us to describe molecules that have double or triple bonds, in addition to molecules with single bonds. We adopt the following definition of multigraph from [Die12] in Definition 5.3.

Definition 5.3: Multigraph (Adopted from [Die12])
A multigraph is a pair (V, E) of disjoint sets of vertices and edges together with a map $E \to V \cup [V]^2$ assigning to every edge either one or two vertices, its ends. Multigraphs can have multiple loops and multiple edges. We can consider a graph to be a multigraph without loops or edges. See Figure 5.3.

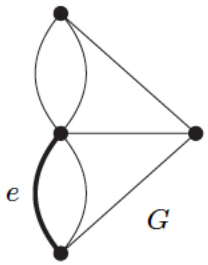


Figure 5.3: A multigraph G with a higlighted edge e. (Figure 1.10.1 in [Die12]).

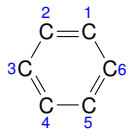
With this notion in mind, we must include the properties of multigraphs, especially multiple edges, in the ontology.

5.2 Examples of Chemical Graphs

To represent adjacency, *adjacency matrices* are normally used to reflect the adjacencies between the atoms in a molecule; they are normally symmetric ($V \times V$, V for the number of vertices). Adjacency matrices are normally used inside the `molfile` format, as it is a common data structure used across chemistry software. Once the adjacency matrix is known, the chemical graph can easily be constructed; a Boolean is stored for every possible pair of atoms ("2" signifies a double bond, "1" signifies a single bond, and "0" if no bond exists). Figure 5.4 outlines an example of the adjacency matrix for benzene (C_6H_6), where the bolded column and row headings indicate the numbered atom in the molecule.

Depending on the chemical database, adjacency matrices may or may not store values for the connected hydrogen atoms; if without, the number of hydrogen atoms can be inferred by counting. Due to space limitations in this thesis, we only present adjacency matrices that are stripped of the hydrogen atoms. As well, we do not consider the other higher powers of adjacency matrices and their eigenvectors – we are primarily interested in the basic connectivity between atoms of molecules, and only use the adjacency matrices.

Consider isooctane, $(CH_3)_3CCH_2CH(CH_3)_2$, as an example. It is also known as "2,2,4-Trimethylpentane" and is an important component of gasoline. In Figure 5.5,



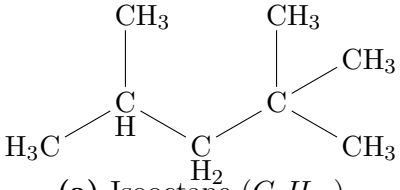
(a) Benzene (C_6H_6; with numbered atoms)

	1	**2**	**3**	**4**	**5**	**6**
1	1	2	0	0	0	1
2	2	1	1	0	0	0
3	0	1	1	2	0	0
4	0	0	2	1	1	0
5	0	0	0	1	1	2
6	1	0	0	0	2	1

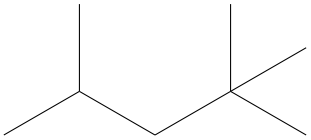
(b) Adjacency matrix for benzene

Figure 5.4: An example of an adjacency matrix. The bolded column and row headings indicate the atom number in (a).

three representations are provided: the molecule itself (with all of the hydrogen atoms), the chemical graph, and the adjacency matrix (stripped of the hydrogens)[2].



(a) Isooctane (C_8H_{18})



(b) Chemical Graph

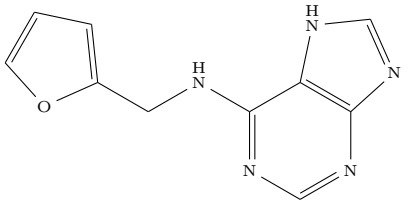
$$A = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

(c) Adjacency Matrix (no hydrogens)

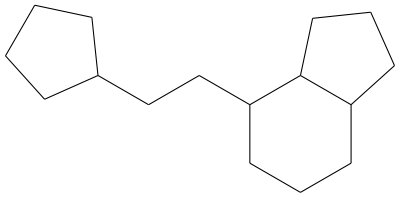
Figure 5.5: Isooctane is represented as a: (a) molecule, (b) chemical graph, and (c) adjacency matrix.

An example of a slightly more complex molecule is kinetin, a class of plant hormone that promotes cell division. It is a type of cytokinin and has the IUPAC name N^6-furfuryladenine and the chemical formula of $C_{10}H_9N_5O$. As we can see in Figure 5.6, this molecule has two fused rings and another ring that is tethered to it. From Figure 5.6(c), we can see how molecules with larger number of atoms will increase the size of the adjacency matrix.

[2]If the hydrogen atoms are included, then the overall adjacency matrix is of size 26 by 26, which would take up most of the page. The condensed version is shown here instead, where we do not include the hydrogen atoms in the adjacency matrix.



(a) Molecule



(b) Chemical Graph

$$A = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 2 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 2 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 2 \\
0 & 0 & 0 & 1 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 1
\end{bmatrix}$$

(c) Adjacency Matrix (no hydrogens)

Figure 5.6: Kinetin is represented as a: (a) molecule, (b) chemical graph, and (c) adjacency matrix (which does not include rows and columns for the hydrogens).

For the rest of this thesis, we are only interested in representing molecules with respect to their chemical graphs. This means that we will *not* present examples of molecules with the hydrogens attached to skeletal backbone.

Before we introduce the axiomatization of chemical graph theory, we will review the basic components of molecules: atoms and bonds. The underlying assumption here is that the reader has a general idea of some of the basic concepts of chemistry taught in high school; in this section, we outline the basic components of molecules.

5.3 Atoms

John Dalton's atomic theory indicates that all matter in the universe is made of tiny particles called atoms. We consider Definition 5.4 and introduce $atom(a)$ in the ontology to represent an atom in a molecule ("a is an atom").

Definition 5.4: Atom
An atom is the smallest particle of an element that cannot be converted into atoms of any other element, nor can they be created, destroyed, or divided.

Atoms of different elements can combine to form compounds. Atoms are made up of subatomic particles known as protons, neutrons, and electrons, where protons and neutrons cluster together to form the nucleus of an atom, while electrons occupy the space that surround the nucleus of an atom. Subatomic particles are associated with electrical charges, as listed in Table 5.2 below.

Table 5.2: Properties of Protons, Neutrons, and Electrons

Subatomic Particle	Charge	Symbol
electron	$1-$	e^-
proton	$1+$	p^+
neutron	0	n^0

Chemists use an *atomic number (Z)* to refer to the number of protons in the nucleus of each atom of an element, and a *mass number (A)* to indicate the total number of protons and neutrons; this is shown in Figure 5.7. *Atomic symbols* (also element symbols) represent each element.

$${}^{A}_{Z}X \qquad \text{Example:} \quad {}^{19}_{9}F$$

Figure 5.7: Atomic number (A), mass number (Z), and atomic symbol (X) of an element; for example, fluorine is denoted by the letter 'F' for its atomic symbol, and has a mass number of 19 and an atomic number is 9.

The modern periodic table shown in Figure 5.8 organizes elements according to their atomic number; according to the *periodic law*, the chemical and physical properties of the elements repeat in a regular, periodic pattern when arranged according to their atomic

numbers. The periodic table's properties can be summarized briefly as follows [Mus01; Mus02]:

- Each element is in a separate box, with its atomic number, atomic symbol, and atomic mass.
- Elements in the table are arranged in 7 periods (rows) and 18 numbered groups (columns).
- Elements in the A groups are the representative elements (also known as main-group elements).
- Elements in the B groups are the transition elements.
- The 'staircase' that runs from the top of Group 13 (IIIA) to the bottom of Group 16 (VIA) separates elements that are classified into three classes: metals, metalloids (semi-metals), and non-metals, as denoted by the colours in the legend of the table.
- Group 1 (IA) are known as *alkali metals*, which react with water to form alkaline or basic solutions.
- Group 2 (IIA) elements are *alkaline earth metals*, which react with oxygen to form *oxides* and can react with water to form alkaline solutions.
- Group 17 (VIIA) elements are known as *halogens*, which can combine with other elements to form salts.
- Group 18 (VIIIA) elements are known as *noble gases*, which do not combine with any other elements.

All elements in each main group have the same number of electrons, known as *valence electrons* – these are responsible for the chemical behaviour of elements. The concept of *bonding* between atoms is a result of the interaction between the valence electrons. The number of valence electrons in an element can be inferred from its group number in the periodic table; for example, Group 2 (IIA) elements have 2 valence electrons. For elements in Groups 13 (IIIA) to 18 (VIIIA), the number of valence electrons is the same as the second digit in the numbering system; for example, elements in Group 15 (VA) have 5 valence electrons, elements in Group 16 (VIA) have 6 valence electrons, and so forth. Valency describes connectivity between elements, usually by determining the number of hydrogen atoms that an element or compound can combine with. Table 5.3 outlines some examples of the valencies for the elements in the listed compounds.

5.4 Bonds

Recall our discussion in Chapter 2 about relations used in [Mag13] and [MKH14]. Since we want to use multigraphs as the basis for our axiomatization, we require our molecular structure ontology to have bonds as *elements* of the domain. In chemical graph theory, edges are relations and not elements of the domain: edges E are relations that only describe pairs of vertices (v_i, v_j) that are connected together; i.e., $(v_i, v_j) \in E$. Furthermore, the existing ontological approaches discussed in Chapter 2 only represent bonds as

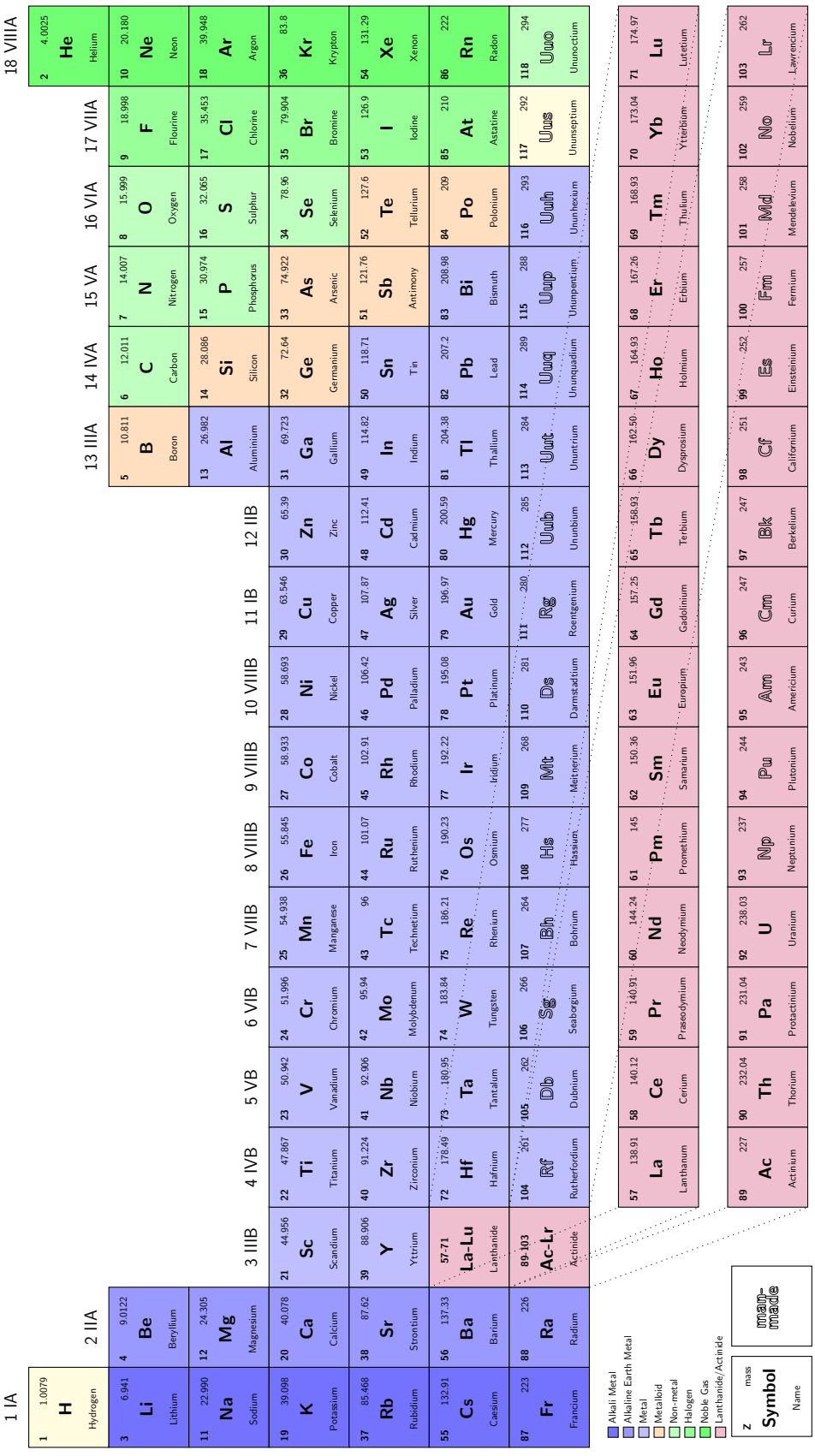
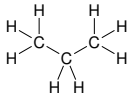
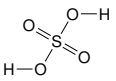
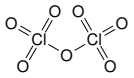


Figure 5.8: Periodic table of elements adapted from Ivan Griffin's LaTeX Periodic Table of Chemical Elements in TikZ [Gri10].

Table 5.3: Examples of valence values found in elements of compounds.

Compound	Diagram	Valencies
Methane CH_4		Carbon: 4 Hydrogen: 1
Propane C_3H_8		Carbon: 4 Hydrogen: 1
Acetylene C_2H_2		Carbon: 4 Hydrogen: 1
Sulfuric Acid H_2SO_4		Sulfur: 6 Oxygen: 2 Hydrogen: 1
Dichlorine Heptoxide Cl_2O_7		Chlorine: 7 Oxygen: 2

relations between two atoms, and rely solely on the atoms as the elements of the domain. For example, rule r_2 in [Mag13] shows that single bonds $singleBond(x, y)$ are binary relations between two atoms x_1 and x_2:

$$singleBond(x_1, x_2) \rightarrow singleBond(x_2, x_1)$$

However, since we want to have an ontology based on multigraphs to properly represent double and triple bonds in chemical graphs, we must adopt the notion that the underlying chemical graph is a multigraph that allows for multiple edges. We cannot utilize these existing ontological approaches to help us axiomatize bonds as they utilize simple graphs that only allow single edges. As we will see in Part II, we will be utilizing graphical incidence structures that are synonymous with multigraphs to verify the ontology; in COLORE, the theory of multigraphs, $T_{multigraphs}$[3], axiomatizes an edge as a class $edge(e)$. Since we will be reusing these multigraph notions in the verification process, we make the design decision to axiomatize these edges (bonds) as *classes* in the ontology.

We consider Definition 5.5, where a *bond* created between two atoms creates a compound that is more stable than the two atoms on their own. We introduce a class called $bond(b)$ in the ontology to signify the concept of a bond ("b is a bond").

Definition 5.5: Bond
A bond is an attraction between two atoms, ions, or molecules that results when there is an electrostatic force of attraction between two oppositely charged ions (ionic bonds), or through the sharing of electrons (covalent bonds).

We introduce the $mol(x, y)$ relation to represent a connection between two components

[3] `http://colore.oor.net/multigraphs/multigraph.clif`

of a molecule ("x is connected to y"). This relation is reflexive and symmetric, as shown in Axioms MG-1 and MG-2[4].

$$\forall x\ (mol(x,x)). \tag{MG-1}$$

$$\forall x \forall y\ (mol(x,y) \supset\ mol(y,x)). \tag{MG-2}$$

Furthermore, as this $mol(x,y)$ relation is intended to demonstrate the connections between two components of a molecule, we emphasize that these components must be of different types; i.e., if used for two atoms $mol(a_1,a_2)$ or two bonds $mol(b_1,b_2)$, then the relation is actually reflexive. We demonstrate this in Axioms MG-3 and MG-4.

$$\forall x \forall y\ ((mol(x,y) \wedge atom(x) \wedge atom(y)) \supset\ (x = y)). \tag{MG-3}$$

$$\forall x \forall y\ ((mol(x,y) \wedge bond(x) \wedge bond(y)) \supset\ (x = y)). \tag{MG-4}$$

A property of chemical bonds is that they must occur between two atoms. Axioms MG-5 indicates that there are at most two atoms in a bond, while Axiom MG-6 indicates that if there is an atom, it must be bonded to another atom and Axiom MG-7 indicates that for any bond that exists, there are two atoms that are in said bond.

$$\begin{aligned}\forall x \forall y \forall z \forall b\ ((atom(x) \wedge atom(y) \wedge atom(z) \wedge bond(b) \wedge \\ mol(x,b) \wedge mol(y,b) \wedge mol(z,b)) \supset \\ ((x = y) \vee (x = z) \vee (y = z))).\end{aligned} \tag{MG-5}$$

$$\begin{aligned}\forall x\ (atom(x) \supset \exists b \exists y((atom(y) \wedge bond(b) \wedge \\ mol(x,b) \wedge mol(y,b)))).\end{aligned} \tag{MG-6}$$

$$\begin{aligned}\forall b\ (bond(b) \supset \exists x \exists y((atom(x) \wedge atom(y) \wedge \\ mol(x,b) \wedge mol(y,b)))).\end{aligned} \tag{MG-7}$$

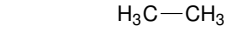
Furthermore, Axioms MG-8 and MG-9 ensure that each bond has at least two atoms, and that these atoms are distinct.

$$\begin{aligned}\forall b\ (bond(b) \supset \exists x \exists y(\forall z_1 \forall z_2(((atom(x) \wedge atom(y) \wedge \\ atom(z_1) \wedge atom(z_2) \wedge mol(z_1,b) \wedge mol(z_2,b)) \supset \\ ((z_1 = x) \vee (z_1 = y) \vee (z_2 = x) \vee (z_2 = y)))))).\end{aligned} \tag{MG-8}$$

[4]We note here that all axioms in this thesis are labelled with a prefix and a number, where the prefixes signify the acronyms for the theory names; e.g., 'MG-1' signifies Axiom #1 in the T_{most_graph} theory of the ontology. We will discuss the various theories of MoSt in greater detail in Part II.

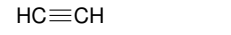
$$\forall b\,(bond(b) \supset\ \exists x \exists y((atom(x) \wedge atom(y) \wedge mol(x,b) \wedge mol(y,b) \wedge (y \neq x)))). \tag{MG-9}$$

Since there are different bond types between atoms, we also make the distinction that any bond can be a single, double, or triple bond. We do not consider quadruple bonds since our primary focus is within organic chemistry (and stable molecules). Figure 5.9 outlines the different bond types found in molecules.



(a) Single bond in ethane (C_2H_6).

(b) Double bond in ethylene (C_2H_4).



(c) Triple bond in acetylene (C_2H_2).

Figure 5.9: Bond types.

Axiom MB-1 describes this statement in first-order logic, where a bond between two distinct atoms is either a single, double, or triple bond.

$$\begin{aligned}\forall b\,(bond(b) \supset \exists x \exists y((atom(x) \wedge atom(y) \wedge (x \neq y) \wedge \\ (singlebond(b,x,y) \vee doublebond(b,x,y) \vee \\ triplebond(b,x,y))))).\end{aligned} \tag{MB-1}$$

Further, we can define the various bond types as follows: a single bond is a bond between two atoms with *at most* one connection between the atoms (Axiom MB-2), a double bond is a bond between atoms with *exactly two* connections (Axiom MB-3), and a triple bond is a bond between atoms with *exactly three* connections (Axiom MB-4).

$$\begin{aligned}\forall b \forall x \forall y\,(singlebond(b,x,y) \equiv (bond(b) \wedge atom(x) \wedge atom(y) \wedge \\ mol(x,b) \wedge mol(y,b) \wedge \forall b_2(((bond(b_2) \wedge \\ mol(x,b_2) \wedge mol(y,b_2)) \supset (b = b_2))))).\end{aligned} \tag{MB-2}$$

$$\begin{aligned}\forall b \forall x \forall y\,(doublebond(b,x,y) \equiv \exists b_2((atom(x) \wedge atom(y) \wedge bond(b_2) \wedge \\ mol(x,b) \wedge mol(y,b) \wedge mol(x,b_2) \wedge \\ mol(y,b_2) \wedge (b \neq b_2)))).\end{aligned} \tag{MB-3}$$

$$\begin{aligned}\forall b \forall x \forall y\,(triplebond(b,x,y) \equiv \exists b_2 \exists b_3((atom(x) \wedge atom(y) \wedge bond(b_2) \wedge \\ bond(b_3) \wedge mol(x,b) \wedge mol(y,b) \wedge mol(x,b_2) \wedge \\ mol(y,b_2) \wedge mol(x,b_3) \wedge mol(y,b_3) \wedge \\ (b \neq b_2) \wedge (b_2 \neq b_3)))).\end{aligned} \tag{MB-4}$$

While it may be simpler to identify bond types between two atoms with a binary relation, there are circumstances where bonds need to be explicitly identified in a molecule. For example, we could have used a binary relation $singlebond(x,y)$ to identify that "there

is a single bond between atoms x and y." However, as we will see in Chapter 7, we need to explicitly specify bonds that 'tether' molecules together, so it is a design choice on our part to have the bond type relations $singlebond(b, x, y)$, $doublebond(b, x, y)$, and $triplebond(b, x, y)$ to be ternary relations instead of binary.

5.5 Elements in the Periodic Table

Keeping in mind the valency of elements, we can also come up with definitions for the notions of elements with a valence of one (single valence), two (double valence), three (triple valence), and four (quadruple valence). Recall Table 5.3 where valence values were listed with the atoms in a compound. The $singlevalence(x)$ relation signifies that "x has a valence of one," $doublevalence(x)$ signifies "x has a valence of two," and so forth. We can axiomatize these classes of valencies as follows:

$$\begin{aligned}\forall x\,(singlevalence(x) \equiv\ & atom(x) \wedge (\forall b_1 \forall b_2 \forall a_1 \forall a_2 (bond(b_1) \wedge bond(b_2) \wedge \\ & mol(a_1, b_1) \wedge mol(x, b_1) \wedge mol(a_2, b_2) \wedge mol(x, b_2) \supset \\ & b_1 = b_2))).\end{aligned} \tag{MED-1}$$

$$\begin{aligned}\forall x\,(doublevalence(x) \equiv\ & (atom(x) \wedge \forall b_1 \forall b_2 \forall b_3 \forall a_1 \forall a_2 \forall a_3 (((atom(a_1) \wedge \\ & atom(a_2) \wedge atom(a_3) \wedge bond(b_1) \wedge bond(b_2) \wedge \\ & bond(b_3) \wedge mol(a_1, b_1) \wedge mol(x, b_1) \wedge mol(a_2, b_2) \wedge \\ & mol(x, b_2) \wedge mol(a_3, b_3) \wedge mol(x, b_3)) \supset ((b_1 = b_2) \vee \\ & (b_1 = b_3) \vee (b_2 = b_3)))))).\end{aligned} \tag{MED-2}$$

$$\begin{aligned}\forall x\,(triplevalence(x) \equiv\ & (atom(x) \wedge \forall b_1 \forall b_2 \forall b_3 \forall b_4 \forall a_1 \forall a_2 \forall a_3 \forall a_4 (((atom(a_1) \wedge \\ & atom(a_2) \wedge atom(a_3) \wedge atom(a_4) \wedge bond(b_1) \wedge \\ & bond(b_2) \wedge bond(b_3) \wedge bond(b_4) \wedge mol(a_1, b_1) \wedge \\ & mol(x, b_1) \wedge mol(a_2, b_2) \wedge mol(x, b_2) \wedge mol(a_3, b_3) \wedge \\ & mol(x, b_3) \wedge mol(a_4, b_4) \wedge mol(x, b_4)) \supset ((b_1 = b_2) \vee \\ & (b_1 = b_3) \vee (b_1 = b_4) \vee (b_2 = b_3) \vee (b_2 = b_4) \vee \\ & (b_3 = b_4)))))).\end{aligned} \tag{MED-3}$$

$$\begin{aligned}\forall x\,(quadvalence(x) \equiv (&atom(x) \wedge \forall b_1 \forall b_2 \forall b_3 \forall b_4 \forall b_5 \forall a_1 \forall a_2 \forall a_3 \forall a_4 \forall a_5 \\ &(((atom(a_1) \wedge atom(a_2) \wedge atom(a_3) \wedge atom(a_4) \wedge \\ &atom(a_5) \wedge bond(b_1) \wedge bond(b_2) \wedge bond(b_3) \wedge \\ &bond(b_4) \wedge bond(b_5) \wedge mol(a_1, b_1) \wedge mol(x, b_1) \wedge \\ &mol(a_2, b_2) \wedge mol(x, b_2) \wedge mol(a_3, b_3) \wedge mol(x, b_3) \wedge \\ &mol(a_4, b_4) \wedge mol(x, b_4) \wedge mol(a_5, b_5) \wedge mol(x, b_5)) \supset \\ &((b_1 = b_2) \vee (b_1 = b_3) \vee (b_1 = b_4) \vee (b_1 = b_5) \vee \\ &(b_2 = b_3) \vee (b_2 = b_4) \vee (b_2 = b_5) \vee (b_3 = b_4) \vee \\ &(b_3 = b_5) \vee (b_4 = b_5)))))).\end{aligned} \tag{MED-4}$$

Additionally, based on the previous section where we indicated that the number of valence electrons corresponds to the group number in the periodic table, we can write axioms to indicate whether an element has a valence value of 1, 2, 3, or 4, as such:

$$\forall x\,(hydrogen(x) \supset singlevalence(x)). \tag{ME-1}$$

$$\forall x\,(oxygen(x) \supset doublevalence(x)). \tag{ME-2}$$

$$\forall x\,(nitrogen(x) \supset triplevalence(x)). \tag{ME-3}$$

$$\forall x\,(carbon(x) \supset quadvalence(x)). \tag{ME-4}$$

Now that we have the basics axiomatized, we can actually go ahead and axiomatize the elements of the periodic table. Take, carbon (C), for example – we can axiomatize it according to how many hydrogen atoms it can bond to. Since we know the valence number is four (4), we can simply indicate in the axiom that there are *at least* four covalent bonds and *at most* four covalent bonds. The axioms for elements are shown in two sentences (*at most* and *at least*) for clarity: when combined together in one axiom, it becomes long and confusing.

$$\begin{aligned}\forall x\,(carbon(x) \supset &atom(x) \wedge (\exists a_1 \exists a_2 \exists a_3 \exists a_4 \exists b_1 \exists b_2 \exists b_3 \exists b_4 (atom(a_1) \wedge \\ &atom(a_2) \wedge atom(a_3) \wedge atom(a_4) \wedge \\ &bond(b_1) \wedge bond(b_2) \wedge bond(b_3) \wedge bond(b_4) \wedge \\ &a_1 \neq b_2 \wedge a_1 \neq b_3 \wedge a_1 \neq b_4 \wedge \\ &a_2 \neq b_1 \wedge a_2 \neq b_3 \wedge a_2 \neq b_4 \wedge \\ &a_3 \neq b_1 \wedge a_3 \neq b_2 \wedge a_3 \neq b_4 \wedge \\ &a_4 \neq b_1 \wedge a_4 \neq b_2 \wedge a_4 \neq b_3 \wedge \\ &mol(a_1, b_1) \wedge mol(x, b_1) \wedge mol(a_2, b_2) \wedge mol(x, b_2) \wedge \\ &mol(a_3, b_3) \wedge mol(x, b_3) \wedge mol(a_4, b_4) \wedge mol(x, b_4)))).\end{aligned} \tag{C-at-least}$$

$$\begin{aligned}\forall x\forall a_1\forall a_2\forall a_3\forall a_4\forall a_5\forall b_1\forall b_2\forall b_3\forall b_4\forall b_5\ (&carbon(x) \wedge atom(a_1)\wedge\\ &atom(a_2) \wedge atom(a_3) \wedge atom(a_4) \wedge atom(a_5)\wedge\\ &bond(b_1) \wedge bond(b_2) \wedge bond(b_3) \wedge bond(b_4) \wedge bond(b_5)\wedge\\ &mol(a_1, b_1) \wedge mol(x, b_1) \wedge mol(a_2, b_2) \wedge mol(x, b_2)\wedge\\ &mol(a_3, b_3) \wedge mol(x, b_3) \wedge mol(a_4, b_4) \wedge mol(x, b_4)\wedge\\ &mol(a_5, b_5) \wedge mol(x, b_5) \supset b_1 = b_2 \vee b_1 = b_3 \vee b_1 = b_4 \vee b_1 = b_5\vee\\ &b_2 = b_3 \vee b_2 = b_4 \vee b_2 = b_5 \vee b_3 = b_4 \vee b_3 = b_5 \vee b_4 = b_5).\end{aligned} \tag{C-at-most}$$

We can write out axioms for each of the elements in a similar manner. Due to the exhaustive list of elements in the periodic table, we only list axioms for common elements that are generally found in small molecules used in drugs; these include hydrogen, nitrogen, and sulfur below:

$$\begin{aligned}\forall x\ (hydrogen(x) \supset atom(x) \wedge (\exists a_1\exists b_1\ (&atom(a_1) \wedge bond(b_1)\wedge\\ &mol(a_1, b_1) \wedge mol(x, b_1)))).\end{aligned} \tag{H-at-least}$$

$$\begin{aligned}\forall x\forall a_1\forall a_2\forall b_1\forall b_2\ (&hydrogen(x)\wedge atom(a_1) \wedge atom(a_2)\wedge\\ &bond(b_1) \wedge bond(b_2)\wedge\\ &mol(a_1, b_1) \wedge mol(x, b_1)\wedge\\ &mol(a_2, b_2) \wedge mol(x, b_2) \supset b_1 = b_2).\end{aligned} \tag{H-at-most}$$

$$\begin{aligned}\forall x(nitrogen(x) \supset atom(x) \wedge (\exists a_1\exists a_2\exists a_3\exists b_1\exists b_2\exists b_3(&atom(a_1)\wedge\\ &atom(a_2) \wedge atom(a_3)\wedge\\ &bond(b_1) \wedge bond(b_2) \wedge bond(b_3)\wedge\\ &a_1 \neq b_2 \wedge a_1 \neq b_3 \wedge a_2 \neq b_1 \wedge a_2 \neq b_3\wedge\\ &a_3 \neq b_1 \wedge a_3 \neq b_2\wedge\\ &mol(a_1, b_1) \wedge mol(x, b_1) \wedge mol(a_2, b_2)\wedge\\ &mol(x, b_2) \wedge mol(a_3, b_3) \wedge mol(x, b_3)))).\end{aligned} \tag{N-at-least}$$

$$\begin{aligned}\forall x\forall a_1\forall a_2\forall a_3\forall a_4\forall b_1\forall b_2\forall b_3\forall b_4\ (&nitrogen(x) \wedge atom(a_1) \wedge atom(a_2)\wedge\\ &atom(a_3) \wedge atom(a_4) \wedge bond(b_1)\wedge\\ &bond(b_2) \wedge bond(b_3) \wedge bond(b_4)\wedge\\ &mol(a_1, b_1) \wedge mol(x, b_1) \wedge mol(a_2, b_2)\wedge\\ &mol(x, b_2) \wedge mol(a_3, b_3) \wedge mol(x, b_3)\wedge\\ &mol(a_4, b_4) \wedge mol(x, b_4) \supset\ b_1 = b_2\vee\\ &b_1 = b_3 \vee b_1 = b_4 \vee b_2 = b_3 \vee b_2 = b_4\vee\\ &b_3 = b_4).\end{aligned} \tag{N-at-most}$$

$$\begin{aligned}
&\forall x\,(sulfur(x) \supset\ atom(x) \wedge (\exists a_1 \exists a_2 \exists a_3 \exists a_4 \exists a_5 \exists a_6 \exists b_1 \exists b_2 \exists b_3 \exists b_4 \exists b_5 \exists b_6 \\
&\quad (atom(a_1) \wedge atom(a_2) \wedge atom(a_3) \wedge atom(a_4) \wedge atom(a_5) \wedge atom(a_6) \wedge \\
&\quad bond(b_1) \wedge bond(b_2) \wedge bond(b_3) \wedge bond(b_4) \wedge bond(b_5) \wedge \\
&\quad bond(b_6) \wedge a_1 \neq b_2 \wedge a_1 \neq b_3 \wedge a_1 \neq b_4 \wedge a_1 \neq b_5 \wedge a_1 \neq b_6 \wedge \\
&\quad a_2 \neq b_1 \wedge a_2 \neq b_3 \wedge a_2 \neq b_4 \wedge a_2 \neq b_5 \wedge a_2 \neq b_6 \wedge a_3 \neq b_1 \wedge \\
&\quad a_3 \neq b_2 \wedge a_3 \neq b_4 \wedge a_3 \neq b_5 \wedge a_3 \neq b_6 \wedge a_4 \neq b_1 \wedge a_4 \neq b_2 \wedge \\
&\quad a_4 \neq b_3 \wedge a_4 \neq b_5 \wedge a_4 \neq b_6 \wedge a_5 \neq b_1 \wedge a_5 \neq b_2 \wedge a_5 \neq b_3 \wedge \\
&\quad a_5 \neq b_4 \wedge a_5 \neq b_6 \wedge a_6 \neq b_1 \wedge a_6 \neq b_2 \wedge a_6 \neq b_3 \wedge a_6 \neq b_4 \wedge \\
&\quad a_6 \neq b_5 \wedge mol(a_1, b_1) \wedge mol(x, b_1) \wedge mol(a_2, b_2) \wedge mol(x, b_2) \wedge \\
&\quad mol(a_3, b_3) \wedge mol(x, b_3) \wedge mol(a_4, b_4) \wedge mol(x, b_4) \wedge mol(a_5, b_5) \wedge \\
&\quad mol(x, b_5) \wedge mol(a_6, b_6) \wedge mol(x, b_6)))).
\end{aligned} \tag{S-at-least}$$

$$\begin{aligned}
&\forall x \forall a_1 \forall a_2 \forall a_3 \forall a_4 \forall a_5 \forall a_6 \forall a_7 \forall b_1 \forall b_2 \forall b_3 \forall b_4 \forall b_5 \forall b_6 \forall b_7\ (sulfur(x) \wedge atom(a_1) \wedge \\
&\qquad atom(a_2) \wedge atom(a_3) \wedge atom(a_4) \wedge atom(a_5) \wedge \\
&\qquad atom(a_6) \wedge atom(a_7) \wedge bond(b_1) \wedge bond(b_2) \wedge \\
&\qquad bond(b_3) \wedge bond(b_4) \wedge bond(b_5) \wedge bond(b_6) \wedge \\
&\qquad bond(b_7) \wedge mol(a_1, b_1) \wedge mol(x, b_1) \wedge mol(a_2, b_2) \wedge \\
&\qquad mol(x, b_2) \wedge mol(a_3, b_3) \wedge mol(x, b_3) \wedge mol(a_4, b_4) \wedge \\
&\qquad mol(x, b_4) \wedge mol(a_5, b_5) \wedge mol(x, b_5) \wedge mol(a_6, b_6) \wedge \\
&\qquad mol(x, b_6) \wedge mol(a_7, b_7) \wedge mol(x, b_7) \supset b_1 = b_2 \vee \\
&\qquad b_1 = b_3 \vee b_1 = b_4 \vee b_1 = b_5 \vee b_1 = b_6 \vee b_1 = b_7 \vee \\
&\qquad b_2 = b_3 \vee b_2 = b_4 \vee b_2 = b_5 \vee b_2 = b_6 \vee b_2 = b_7 \vee \\
&\qquad b_3 = b_4 \vee b_3 = b_5 \vee b_3 = b_6 \vee b_3 = b_7 \vee b_4 = b_5 \vee \\
&\qquad b_4 = b_6 \vee b_4 = b_7 \vee b_5 = b_6 \vee b_5 = b_7 \vee b_6 = b_7).
\end{aligned} \tag{S-at-most}$$

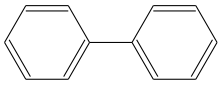
The rest of the axiomatizations for the elements can be found online in COLORE in individual CLIF files[5].

5.6 Reasoning With These Axioms

While having this axiomatization of molecular graph theory gives us the ability to do first-order reasoning, it is *insufficient*. Recall our earlier discussion in Chapter 2.4 – molecular graph theory does not consider functional groups as elements of the domain. Our current axiomatization does not allow us to identify such functional groups, nor can we be able to answer any of the competency questions that were presented in Chapter 3.2. For example, in a molecule such as biphenyl ($C_{12}H_{10}$), our current axioms are unable to allow us to identify one of the benzene rings as a group, nor can we make any distinctions between the two benzene rings in Figure 5.10. At most, the only thing we can do is outline

[5] `http://colore.oor.net/molecular_graph/definitions/most_elements.clif`

how each individual atom in the molecule is connected to another atom.



(a) Biphenyl ($C_{12}H_{10}$)

(b) Can you distinguish between the blue and purple rings with our current axiomatization?

Figure 5.10: Biphenyl ($C_{12}H_{10}$)

This is very similar to the work done in [Mag13] and [MKH14], where a rule-based approach is utilized to enumerate through all of the atoms and bonds in a molecule. The primary approach taken in both was more geared toward classification using DL, particularly the OWL and Semantic Web Rule Language (SWRL) languages, and to show whether it is possible to represent cyclic structures using nonmonotonic existential rules. In their OWL axiomatization, the primary focus here is classifying the atoms in cyclic structures, along with applying rules to improve the computational properties of running these rules in a DL reasoner.

While the work done [Mag13] and [MKH14] is geared with a computational approach in mind, we realize that this approach to using DL and OWL is not as intuitive as the axiomatization we present in the next chapter. The axioms presented in this chapter are only sufficient to describe molecules on an atom-by-atom basis, but do not provide any additional insight, such as which atoms constitute a functional group, or whether groups are rings, and so forth. What we need are some additional relations and axioms that extend the axioms presented in this chapter, particular the notions of rings, chains, and functional groups that are discussed further in the next chapter.

5.7 Summary

In this chapter, we have presented axioms for the basic notions found in chemical graph theory: atoms and bonds are connected together, and are represented as vertices and edges in a graph. We have added the notion of a multigraph in the axiomatization of chemical graph theory to allow for graphs to have multiple edges. With respect to the requirements presented in Chapter 3, we have partially satisfied Requirements (**R-1**), (**R-2**), and (**R-3**) since elements of the domain are currently atoms and bonds, and we are utilizing multigraphs as the basis of the ontology. Next, we will discuss how we extend these simple axioms with the notions of functional groups.

Chapter 6

Rings, Chains & Functional Groups

In this chapter, we discuss the notions of rings, chains, and functional groups in chemistry, along with several common-sense intuitions we have of these concepts, and discuss how we properly axiomatize these concepts in first-order logic as elements of the domain.

6.1 Rings

In chemistry, the term *ring* is used to identify a simple cycle of atoms and bonds in a molecule, or a connected set of atoms and bonds in which every atom and bond is a member of the cycle (also referred to as a *ring system*, defined in Definition 6.1).

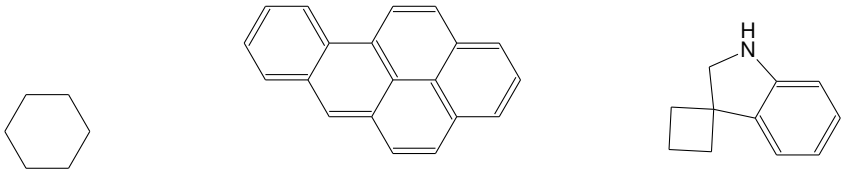
Definition 6.1: Ring (System)
A ring is a simple cycle of atoms and bonds.

A ring system where there is only one simple cycle is referred to as a monocycle or a simple ring. In contrast, a ring system that is not a simple cycle is called a polycycle or a polycyclic ring system as defined in Definition 6.2; for compounds that consist of more than one element, they are also referred to as heterocyclic (see Definition 6.3). In Figure 6.1, we show an example of compounds that are considered as monocyclic and polycyclic; Cyclohexane is considered monocyclic, while benzo[a]pyrene and 1',2'-Dihydrospiro[cyclobutane-1,3'-indole] are considered homocyclic and heterocyclic molecules, respectively.

Definition 6.2: Polycylic Compound
A compound consisting of a ring system that has more than one simple cycle.

Definition 6.3: Heterocyclic Compound (Adopted from [IUP06])
A heterocyclic compound consists of atoms of at least two different elements as members of its rings.

For molecules that contain one or more rings, these are referred to as *cyclic compounds*, and molecules consisting of two or more rings are referred to as *polycyclic compounds*. Further, molecules that contain no rings are referred to as *acyclic* or open-chain



(a) Cyclohexane (C_6H_{12})

(b) Benzo[a]pyrene ($C_{20}H_{12}$)

(c) 1',2'-Dihydrospiro[cyclobutane-1,3'-indole] ($C_{11}H_{13}N$)

Figure 6.1: Examples of monocyclic and polycyclic compounds.

compounds. Examples of cyclic molecules include: cyclopropane (C_3H_6), cyclobutane (C_4H_8), cyclopentane (C_5H_{10}), and cyclohexane (C_6H_{12}). In Figure 6.2, we show examples of various cycloalkenes: the first row of molecular diagrams include the carbon and hydrogen atoms, and the bottom row indicate the corresponding skeletal diagrams.

(a) C_3H_6 **(b)** C_4H_8 **(c)** C_5H_{10} **(d)** C_6H_{12}

(e) Cyclopropane **(f)** Cyclobutane **(g)** Cyclopentane **(h)** Cyclohexane

Figure 6.2: Examples of cyclicalkenes; the top row outlines the molecular diagrams with carbon and hydrogen atoms labelled, and the bottom row of outlines the corresponding skeletal diagrams for these compounds.

6.2 Chains

Chains, on the other hand, do not involve any looping or connections between one end of the chain to the other; they are often referred to as *open-chain compounds* or *acyclic compounds*.

Definition 6.4: Chain
A chain is the bonding of atoms, usually of the same element, into a series. A chain may or may not have any side chains attached to it.

In organic chemistry and biochemistry, chemical groups that are attached to the backbone of a molecule are referred to as *side-chains*. Usually these are found in amino acids: each

amino acid has a different side chain that is attached to the central carbon atom; R-groups vary from a single hydrogen atom to large structures containing carbon atoms linked into rings [Fre+16]. The general structure of an amino acid is shown in Figure 6.3, where a central carbon atom (known as $\alpha-$carbon) bonds covalently to four different atoms or groups of atoms:

1. A hydrogen atom (H),
2. An amino group (NH_2),
3. A carboxyl group ($COOH$), and
4. A distinctive side chain (also known as a R group)

Figure 6.4 outlines twenty amino acids commonly found in proteins, each with a different R-group that determines its chemical nature. Examples of R-groups include $—CH_3$ for alanine, $—CH_2CH_2CONH_2$ for glutamine, and $—CH_2OH$ for serine.

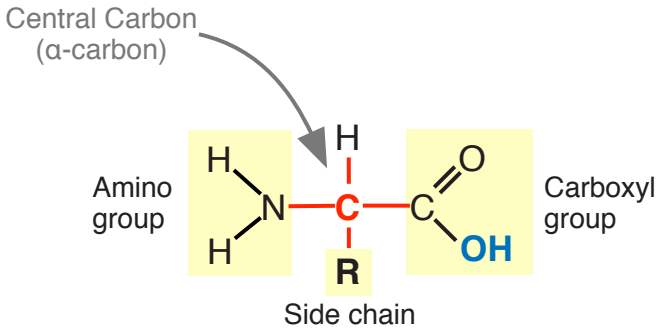


Figure 6.3: Basic structure of an amino acid.

For compounds that have no side chains, they are called *straight-chain compounds* ('straight' in the sense that they are drawn to be schematically straight); in contrast, there are also compounds called *branched-chains* where compounds that have more than four carbons can have structures that branch. Figure 6.5 shows a comparison of these chain molecules.

To properly describe straight-chains and branched-chains, we introduce two notions to help us describe these characteristics of straight- and branched-chains: *ends* and *forks*. We can think of chains as an acyclic collection of atoms bonded together, where there is an atom at the start of the chain and an atom at the end of the chain. We can define an *end atom* to be an atom that is at either end of the chain and is only bonded to at most one other atom, as shown in Definition 6.5.

Definition 6.5: End Atom
An end atom is at one end of a straight-chain compound, and is in at most one bond in a functional group.

In first-order logic, we can define this as Axiom MD-1, where an end atom is in a func-

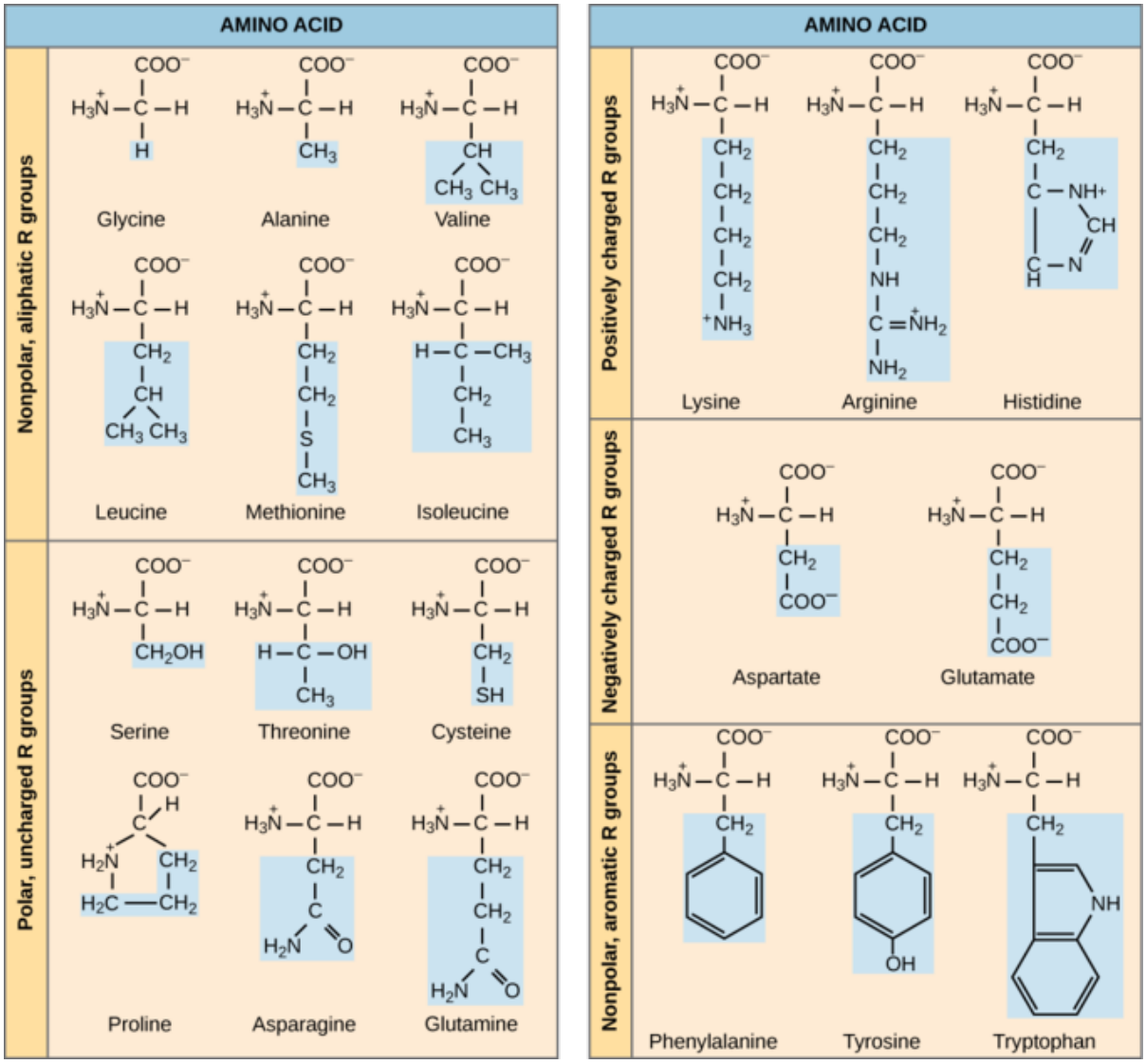


Figure 6.4: Major amino acids found in organisms. The R groups (side chains) are highlighted for each amino acid. (Image from [The17]).

(a) Isopentane (C_5H_{12}) **(b)** Pentane (C_5H_{12})

Figure 6.5: Examples of open-chained compounds: isopentane (C_5H_{12}) is a structural isomer of pentane (C_5H_{12}) where the former is a branched-chain compound, and the latter is a straight-chain compound.

tional group that is *only* bonded to one other atom within the group.

$$\begin{aligned}\forall x \forall y\,(end(x,y) \equiv (&atom(x) \wedge group(y) \wedge mol(x,y) \wedge \\ &\forall b_1 \forall b_2 \forall w \forall z(((bond(b_1) \wedge bond(b_2) \wedge \\ &atom(w) \wedge atom(z) \wedge mol(x,b_1) \wedge \\ &mol(z,b_1) \wedge mol(x,b_2) \wedge mol(w,b_2) \wedge \\ &mol(z,y) \wedge mol(w,y)) \supset (w = z))))).\end{aligned} \tag{MD-1}$$

Conversely, we can think of a *fork atom* as an atom that is bonded to at least three atoms – an analogy would be a 'fork' in a road that leads to three (or more) paths. We do not use the term 'branch' to describe this atom since it may contain additional connotations (for chemists) that may not become axiomatized in the ontology.

Definition 6.6: Fork Atom (Branching Atom)
A fork atom is in a functional group and is bonded to at least three other atoms. We can also call this a branching atom.

As is defined in Axiom MD-2, a fork atom is bonded to at least three other atoms that are not necessarily in a group.

$$\begin{aligned}\forall x \forall y\,(fork(x) \equiv (&atom(x) \wedge \exists b_1 \exists b_2 \exists b_3 \exists a_1 \exists a_2 \exists a_3((atom(a_1) \wedge \\ &atom(a_2) \wedge atom(a_3) \wedge bond(b_1) \wedge bond(b_2) \wedge \\ &bond(b_3) \wedge (a_1 \neq a_2) \wedge (a_2 \neq a_3) \wedge (a_1 \neq a_3) \wedge \\ &(b_1 \neq b_2) \wedge (b_2 \neq b_3) \wedge (b_1 \neq b_3) \wedge \\ &mol(x,b_1) \wedge mol(a_1,b_1) \wedge mol(x,b_2) \wedge \\ &mol(a_2,b_2) \wedge mol(x,b_3) \wedge mol(a_3,b_3))))).\end{aligned} \tag{MD-2}$$

Consider the examples presented in Figure 6.5, we show which atoms are the ends and forks in Figure 6.6.

(a) Isopentane (C_5H_{12}) **(b)** Pentane (C_5H_{12})

Figure 6.6: Examples of forks and ends in isopentane and pentane, highlighted in red and blue, respectively.

With these two distinctions, we can define *straight-chained* compounds as functional groups that *do not* have ends and forks, as outlined in Axiom MD-3. Pentane in Figure 6.6(b) is also an example of a straight chain. For the rest of this dissertation, when

'chain' is mentioned, we mean straight-chained compounds.

$$\forall x\,(chain(x) \equiv (group(x) \wedge \exists y((end(y,x) \wedge \forall w(((atom(w) \wedge mol(w,x)) \supset \neg fork(w))))))). \tag{MD-3}$$

Recall Definition 6.1, where we consider rings as simple cycles that contain atoms and bonds. With the definitions of *end* and *forks* in mind, we can state that rings are functional groups that do not contain atoms that are ends and forks. We can define this notion of ring in first-order logic as follows (see Axiom MD-4 and Figure 6.7):

$$\forall x\,(ring(x) \equiv (group(x) \wedge \forall y(((atom(y) \wedge mol(y,x)) \supset (\neg end(y,x) \wedge \neg fork(y)))))). \tag{MD-4}$$

Now, we can move onto discussing functional groups and the role they play in molecules.

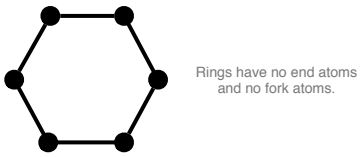


Figure 6.7: A ring does not contain any end or fork atoms.

6.3 Functional Groups

Having discussed the types of shapes found in compounds, we are primarily interested in groups of atoms in molecules. These *functional groups*, as defined in Definition 6.7, are specific groups of atoms that have very characteristic properties regardless of the other atoms present in the molecule [Mus02]; for example, these include alcohols, amines, carboxylic acids, ketones, and ethers (see Tables 6.1 and 6.2).

Definition 6.7: Functional Group
A functional group is a group of bonded atoms in an organic compound that provides compounds with characteristic properties.

Chemically, a functional group behaves in the same way in every molecule it is part of: the chemistry of every molecule, regardless of its size and complexity, is determined by the functional groups it contains [McM15]. Behaviours of compounds are dictated by groups of hydrogen H, nitrogen N, oxygen O, phosphorous P, or sulfur S atoms that are bonded to one of the carbon atoms in a specific way [Fre+16]. The prominent functional groups found in organic molecules and recognized by chemists are summarized in Tables 6.1 and 6.2:

- *Amino* and *carboxyl* functional groups attract or release a hydrogen ion (proton), respectively, when in a solution [Fre+16]. This means that amino groups function as bases and carboxyl groups act as acids.

- *Carbonyl* functional groups normally link molecules such as formaldehyde, acetaldehyde, and acetone into larger complex compounds.
- *Hydroxyl* functional groups act as weak acids and are usually soluble in water.
- *Phosphate* functional groups are negatively charged and often dramatically affect the structure of the recipient molecule when bonded.
- *Sulfhydryl* functional groups consist of a sulfur atom that is bonded to a hydrogen, allowing for the potential to bond with other sulfhydryl groups via disulfide (S—S) bonds.

Furthermore, we also make the distinctions for the following notions found in functional groups: saturated, unsaturated, alternating, and trivial groups. For functional groups that *only* have single bonds, we can classify them as *saturated*; we define this in Definition 6.8 and in Axiom MD-5.

Definition 6.8: Saturated Functional Group
A saturated functional group is a group that only has single bonds.

$$\begin{aligned}\forall x\,(saturated(x) \equiv (group(x) \wedge \forall b \forall y \forall z(((bond(b) \wedge atom(y) \wedge \\ mol(y,b) \wedge mol(z,b) \wedge mol(y,x) \wedge \\ mol(z,x)) \supset singlebond(b,y,z))))).\end{aligned} \tag{MD-5}$$

Conversely, for functional groups that contain at least one double bond[1], we can classify them as unsaturated; we define this in Definition 6.9 and in Axiom MD-6. In Table 6.3, cyclobutane is saturated while cyclobutene is unsaturated.

Definition 6.9: Unsaturated Functional Group
An unsaturated functional group is a group that has at least one double bond.

$$\begin{aligned}\forall x\,(unsaturated(x) \equiv (group(x) \wedge \exists b \exists y \exists z(((atom(y) \wedge bond(b) \wedge \\ mol(y,b) \wedge mol(z,b) \wedge mol(y,x) \wedge \\ mol(z,x)) \supset doublebond(b,y,z))))).\end{aligned} \tag{MD-6}$$

There are also some compounds and functional groups where the bonds alternate between single and double, we call these alternating groups; we define this in Definition 6.10 and in Axiom MD-7. Examples of molecules that have alternating single and multiple bounds would be benzene (C_6H_6) and cyclooctatetraene (C_8H_8) shown in Figure 6.8.

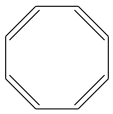
Definition 6.10: Alternating Functional Groups
An alternating functional group is a group with bonds that alternate from

[1] We have chosen to only include double bonds in this definition, but the implication is that this also applies for triple and quadruple bonds.

single and double.

$$\forall x \forall y\ (alternatinggroup(x) \equiv ((group(x) \wedge atom(y) \wedge mol(x,y)) \supset \exists a_1 \exists a_2 \exists b_1 \exists b_2((atom(a_1) \wedge atom(a_2) \wedge bond(b_1) \wedge bond(b_2) \wedge mol(y,b_1) \wedge mol(a_1,b_1) \wedge doublebond(b_2,y,a_2) \wedge mol(a_1,x) \wedge mol(a_2,x))))). \quad \text{(MD-7)}$$

(a) Benzene (C_6H_6)



(b) Cyclooctatetraene (C_8H_8)

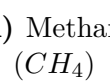
Figure 6.8: Examples of alternating single and multiple bonds.

There are some circumstances where a single atom (without the associated hydrogens) may be considered as a functional group, such as methane (CH_4) and ammonia (NH_3), where we have shown the skeletal formulae in Figure 6.9; we define this as a trivial group in Definition 6.11 and in Axiom MD-8. We also note that a trivial group satisfies the definition of a chain since it contains one end atom.

Definition 6.11: Trivial Group
A trivial functional group is a functional group that only contains a single atom.

$$\forall x\ (trivialgroup(x) \equiv (group(x) \wedge \forall y \forall z(((atom(y) \wedge atom(z) \wedge mol(y,x) \wedge mol(z,x)) \supset (y = z))))). \quad \text{(MD-8)}$$

(a) Methane (CH_4)



(b) Methane as a trivial group

(c) Ammonia (NH_3)

(d) Ammonia as a trivial group

Figure 6.9: Examples of trivial groups. The blue '1' in (b) and (d) the numbering of atoms.

Table 6.1: Examples Functional Groups. (Adapted from [Fre+16].)

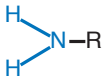
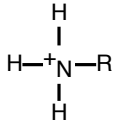
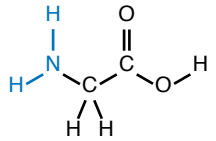
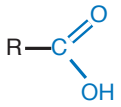
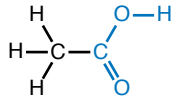
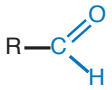
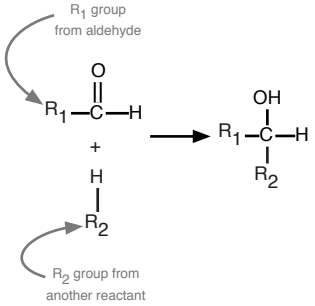
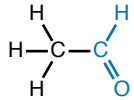
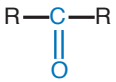
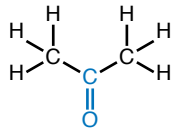
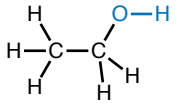
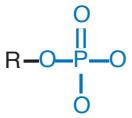
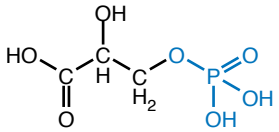
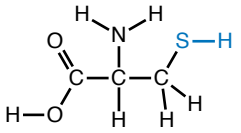
Functional Group	Formula	Family of Molecules	Properties	Example
Amino		Amines	Acts as a base and attracts protons to form: 	 Glycine
Carboxyl		Carboxylic Acids	Acts as an acid and loses a proton in solution to form: 	 Acetic Acid
Carbonyl		Aldehydes	Aldehydes react with compounds to produce larger molecules of the following form: 	 Acetylaldehyde
		Ketones		 Acetone

Table 6.2: Examples Functional Groups (cont.). (Adapted from [Fre+16].)

Functional Group	Formula	Family of Molecules	Properties	Example
Hydroxyl		Alcohols	Makes compounds more soluble through hydrogen bonding with water.	 Ethanol
Phosphate		Organic Phosphates	Molecules with more than one phosphate linked together can store large amounts of chemical energy.	 3-Phosphoglyceric Acid
Sulfhydryl		Thiols	Form disulfide (S—S) bonds that contribute to protein structure	 Cysteine

Consider functional groups of atoms that do not include any forks or complex, aggregated compounds. In Definition 6.12, we make the distinction between the notion of a functional group and primitive functional group. Functional groups can be composed of other functional groups, but we make the distinction in the ontology where *primitive* functional groups form the basis of other functional groups and compounds; these are groups that cannot be reduced any further into additional groups. Examples of *primitive* functional groups are shown in Table 6.3 (not exhaustive) – notice how all the groups listed have skeletal formulae that consist of only (straight-)chains or rings.

Definition 6.12: Primitive Functional Group
A primitive functional group is a functional group that cannot be reduced to one or more functional groups.

6.4 General Intuitions of Rings, Chains, and Functional Groups

Now that we have outlined the basic concepts of molecular structure, we can outline commonsense intuitions of rings, chains, and functional groups. The first and foremost intuition is that all atoms, bonds, and groups are disjoint elements in the ontology; for example, all atoms are not bonds, all bonds are not groups, and all atoms are not groups, and so forth – this is expressed in Axioms M-1 and M-2.

$$\forall x\ (atom(x) \supset \neg(bond(x) \vee group(x))). \tag{M-1}$$

$$\forall x\ (bond(x) \supset \neg group(x)). \tag{M-2}$$

Additionally, similar to Axioms MG-3 and MG-4 in the previous chapter, we only allow components of different types for the $mol(x, y)$ relation, and address this as well for functional groups in Axiom M-3.

$$\forall x \forall y\ ((mol(x, y) \wedge group(x) \wedge group(y)) \supset (x = y)). \tag{M-3}$$

For every functional group, we can also specify the following intuitions:

- In Axiom M-4, we specify that there is at least one atom in the functional group.

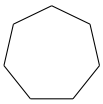
$$\forall x\ (group(x) \supset \exists a((atom(a) \wedge mol(a, x)))). \tag{M-4}$$

- In Axiom M-5, every atom is in a functional group. This ensures that there are no functional groups that have *no* atoms.

$$\forall x\ (atom(x) \supset \exists y((group(y) \wedge mol(x, y)))). \tag{M-5}$$

- In Axiom M-6, every atom has a bond with another atom in a functional group.

Table 6.3: Examples of Primitive Functional Groups

Primitive Functional Group	**Structure**
Ethane (C_2H_6)	
Ethylene (C_2H_4)	
Acetylene (C_2H_2)	
Cyclopropane (C_3H_6)	
Cyclobutane (C_4H_8)	
Cyclobutene (C_4H_6)	
Cyclopentane (C_5H_{10})	
Cyclohexane (C_6H_{12})	
Cycloheptane (C_7H_{14})	
Cyclopentadiene (C_5H_6)	
Benzene (C_6H_6)	

This is to also ensure that there do not exist any atoms that are not bonded with anything in a functional group.

$$\forall x\,(atom(x) \supset \exists b \exists y \exists z((bond(b) \wedge group(y) \wedge atom(z) \wedge (x \neq z) \wedge mol(x,b) \wedge mol(z,b) \wedge mol(z,y)))). \tag{M-6}$$

- In Axiom M-7, we also have transitivity between atoms and bonds in a functional group: any bond in a group implies that the atoms are also in the group.

$$\forall b \forall g \forall x \forall y\,((atom(x) \wedge atom(y) \wedge (x \neq y) \wedge bond(b) \wedge group(g) \wedge mol(x,b) \wedge mol(y,b) \wedge mol(b,g)) \supset mol(x,g)). \tag{M-7}$$

- Similarly, in Axiom M-8, we can also state that, if an atom is in a bond that is in a functional group, then the atom is also in the group.

$$\forall x \forall y \forall z\,((atom(x) \wedge bond(y) \wedge group(z) \wedge mol(x,y) \wedge mol(y,z)) \supset mol(x,z)). \tag{M-8}$$

Additionally, the following commonsense intuitions are explicitly axiomatized in the ontology:

- In Axiom M-9, there is at least one atom in a group.

$$\forall y\,(group(y) \supset \exists x((atom(x) \wedge mol(x,y)))). \tag{M-9}$$

- In Axiom M-10, if two atoms are in a group and have a bond between each other, then the bond is also in the group. Suppose there are two bonds b_1 and b_2 that have the same atoms a_1 and a_2, such that b_1 is in group g_1. Both atoms must be in both groups, therefore both bonds must be in the same group.

$$\forall x \forall y \forall b \forall g\,((atom(x) \wedge atom(y) \wedge bond(b) \wedge group(g) \wedge (x \neq y) \wedge mol(x,g) \wedge mol(y,g) \wedge mol(x,b) \wedge mol(y,b)) \supset mol(b,g)). \tag{M-10}$$

- In Axiom M-11, if there is a bond b that is not in a group g, but there exists an atom a in the bond b, then the atom a is also not in the group g.

$$\forall b \forall g\,((bond(b) \wedge group(g) \wedge \neg mol(b,g)) \supset \exists a((atom(a) \wedge mol(a,b) \wedge \neg mol(a,g)))). \tag{M-11}$$

Finally, we can relate the properties of functional groups with the notions of ends and forks presented in Axioms MD-1 and MD-2, respectively, with the following axioms:

- Recall that we focus only on *primitive* functional groups, so it is suffice to state in Axiom M-12 that a functional group only has at most two ends. Having more than

two ends indicates that there is a fork in compound, which violates our notion of a primitive group.

$$\forall x \forall y \forall z \forall w \, ((group(x) \wedge end(y,x) \wedge end(z,x) \wedge end(w,x)) \supset ((y=z) \vee (y=w) \vee (z=w))). \tag{M-12}$$

- Furthermore, we also include an axiom to indicate that an atom in a molecule is *bonded* with two other atoms in Axiom M-13. Two atoms are *bonded* if there exists a bond that the atoms are in.

$$\begin{aligned} \forall a_1 \forall a_2 \forall a_3 \forall a_4 \forall g \forall b_1 \forall b_2 \forall b_3 \, (&group(g) \wedge atom(a_1) \wedge atom(a_2) \wedge atom(a_3) \wedge \\ &atom(a_4) \wedge mol(a_1,g) \wedge mol(a_2,g) \wedge \\ &mol(a_3,g) \wedge mol(a_4,g) \wedge bond(b_1) \wedge \\ &mol(b_1,g) \wedge mol(a_1,b_1) \wedge mol(a_2,b_1) \wedge \\ &bond(b_2) \wedge mol(b_2,g) \wedge mol(a_1,b_2) \wedge \\ &mol(a_3,b_2) \wedge bond(b_3) \wedge mol(b_3,g) \wedge \\ &mol(a_1,b_3) \wedge mol(a_4,b_3) \supset \ (a_4=a_3) \vee \\ &(a_2=a_3) \vee (a_4=a_2)). \end{aligned} \tag{M-13}$$

With these notions of forks and ends in mind, we can also specify the Theorem 6.4.1: for a fork atom that is in a group, there exists an atom and a bond that is not part of that group.

Theorem 6.4.1: Fork Atoms
For a fork atom that is in a group, there exists an atom and a bond that is not part of that group.

$$\begin{aligned} \forall x \forall g \, (fork(x) \wedge mol(x,g) \wedge group(g)) \supset \exists b \exists y \, &atom(y) \wedge bond(b) \wedge \\ &mol(x,b) \wedge mol(y,b) \wedge \neg mol(y,g). \end{aligned} \tag{M-ThF}$$

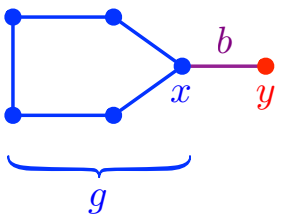


Figure 6.10: Theorem for fork atoms. The group g is highlighted in blue, the fork atom is labelled as x in blue, the bond between the fork atom and another atom is labelled as b in purple, and the third atom y is highlighted in red. We can see that y is not in the group g.

This is graphically depicted in Figure 6.10, where the group g is highlighted in blue and the fork atom is labelled as x in blue. The bond between the fork atom and another atom y is labelled as b in purple, and the third atom y is highlighted in red. We can see that y is not in the group g. This should hold for every group that contains a fork atom;

the proof for this can be found in Appendix D.1 and online in COLORE[2]. This theorem is especially important in the context of MoSt since it is a *nontrivial* sentence that is entailed by the ontology – this property of fork atoms has nothing to do with specific molecules, as it is a general property that applies to all fork atoms.

6.5 Summary

From what we've presented in this chapter, we can axiomatize the basic structures of molecules by identifying the functional groups and the bonds between the atoms that form the groups.

We can axiomatize individual functional groups, such as those presented in Table 6.3; for example, we show the axiomatizations for ethane, ethylene, and cyclopropane, and benzene in Figures 6.11 to 6.14, respectively. We call these definitions "molecular descriptions," which we provide a more formal definition in Chapter 15, but for the purposes of Part I of this work, we can simply think of these definitions as axioms of the ontology that are used to define functional groups.

In Figure 6.11, we can see that this is one of the simplest chains that can be axiomatized in the ontology: ethane consists of two carbon atoms bonded together by a single bond. In the axiom, we can see that we have explicitly defined that $ethane(x)$ is a chain $chain(x)$, is saturated $saturated(x)$, and that there exists two carbon atoms $carbon(c_1)$ and $carbon(c_2)$ which are unique ($c_1 \neq c_2$), and is single-bonded by a bond b: $singlebond(b, c_1, c_2)$. The closure axiom at the end $\forall y\, mol(y, x) \equiv ((y = c_1) \vee (y = c_2))$ indicates that the only possible atoms in this chain are the carbon atoms.

$$\begin{aligned} \forall x\, ethane(x) \equiv\ & chain(x) \wedge saturated(x) \wedge \exists c_1 \exists c_2 \exists b\, carbon(c_1) \wedge \\ & carbon(c_2) \wedge (c_1 \neq c_2) \wedge singlebond(b, c_1, c_2) \wedge \\ & (\forall y\, mol(y, x) \equiv ((y = c_1) \vee (y = c_2))) \end{aligned}$$

—

Figure 6.11: Axiomatization of ethane (C_2H_6).

Similarly with ethylene in Figure 6.12, the axiom is more or less the same with the exception that the bond between the carbon atoms c_1 and c_2 is a double bond, not a single bond: $doublebond(b, c_1, c_2)$. For functional groups that are cyclic, the axioms are

$$\begin{aligned} \forall x\, ethylene(x) \equiv\ & chain(x) \wedge unsaturated(x) \wedge \exists c_1 \exists c_2 \exists b\, carbon(c_1) \wedge \\ & carbon(c_2) \wedge (c_1 \neq c_2) \wedge doublebond(b, c_1, c_2) \wedge \\ & (\forall y\, mol(y, x) \equiv ((y = c_1) \vee (y = c_2))) \end{aligned}$$

==

Figure 6.12: Axiomatization of ethylene (C_2H_4).

defined in a similar manner, where it is explicitly stated that the group is a ring $ring(x)$

[2] http://colore.oor.net/most/interprets/fork_theorem/

and whether it is saturated $saturated(x)$ or unsaturated $unsaturated(x)$, along with an enumeration of the atoms and bonds found in the group. In the case of Figure 6.13, cyclopropane consists of three carbon atoms that are single bonded together. For benzene in Figure 6.14, we can see that benzene consists of both single and double bonds, so it is unsaturated $unsatured(x)$ and contains a much longer enumeration of atoms and bonds (since there are six carbon atoms, three single bonds, and three double bonds).

$$\begin{aligned}\forall x\, cyclopropane(x) \equiv\ & ring(x) \wedge saturated(x) \wedge \exists c_1 \exists c_2 \exists c_3 \exists b_1 \exists b_2 \exists b_3 \\ & carbon(c_1) \wedge carbon(c_2) \wedge carbon(c_3) \wedge \\ & (c_1 \neq c_2) \wedge (c_1 \neq c_3) \wedge (c_2 \neq c_3) \wedge \\ & (b_1 \neq b_2) \wedge (b_1 \neq b_3) \wedge (b_2 \neq b_3) \wedge \\ & singlebond(b_1, c_1, c_2) \wedge singlebond(b_2, c_2, c_3) \wedge \\ & singlebond(b_3, c_3, c_1) \\ & (\forall y\, mol(y, x) \equiv ((y = c_1) \vee (y = c_2) \vee (y = c_3))\end{aligned}$$

Figure 6.13: Axiomatization of cyclopropane (C_2H_4).

$$\begin{aligned}\forall x\, benzene(x) \equiv\ & ring(x) \wedge unsaturated(x) \wedge \exists c_1 \exists c_2 \exists c_3 \exists c_4 \exists c_5 \exists c_6 \\ & \exists b_1 \exists b_2 \exists b_3 \exists b_4 \exists b_5 \exists b_6\ carbon(c_1) \wedge carbon(c_2) \wedge \\ & carbon(c_3) \wedge carbon(c_4) \wedge carbon(c_5) \wedge \\ & carbon(c_6) \wedge (c_1 \neq c_2) \wedge (c_1 \neq c_3) \wedge (c_1 \neq c_4) \wedge \\ & (c_1 \neq c_5) \wedge (c_1 \neq c_6) \wedge (c_2 \neq c_3) \wedge (c_2 \neq c_4) \wedge \\ & (c_2 \neq c_5) \wedge (c_2 \neq c_6) \wedge (c_3 \neq c_4) \wedge (c_3 \neq c_5) \wedge \\ & (c_3 \neq c_6) \wedge (c_4 \neq c_5) \wedge (c_4 \neq c_6) \wedge (c_5 \neq c_6) \wedge \\ & (b_1 \neq b_2) \wedge (b_1 \neq b_3) \wedge (b_2 \neq b_3) \wedge \\ & singlebond(b_1, c_1, c_2) \wedge doublebond(b_2, c_2, c_3) \wedge \\ & singlebond(b_3, c_3, c_4) \wedge doublebond(b_4, c_4, c_5) \wedge \\ & singlebond(b_5, c_5, c_6) \wedge doublebond(b_6, c_6, c_1) \wedge \\ & (\forall y\, mol(y, x) \equiv ((y = c_1) \vee (y = c_2) \vee (y = c_3) \vee \\ & (y = c_4) \vee (y = c_5) \vee (y = c_6)))\end{aligned}$$

Figure 6.14: Axiomatization of benzene (C_2H_4).

Components of a Functional Group Definition Notice how each functional group definition consists of the following components:

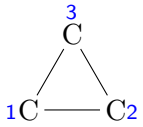
- Identification of whether the functional group is a chain or ring,
- Whether the group is saturated or unsaturated,

- A listing of all the atoms found in the group,
- Inequalities for all the atoms – for example, we do not want individual carbon atoms to all be the same,
- Identification of the types of bonds between the atoms, and
- A *closure statement* to ensure that only the atoms listed are contained within the group.

Closure Axioms The *closure axiom* is especially important in the axiomatization of functional groups as it ensures that only the elements listed are found in the group, and are the only elements that are *connected* in the group via the $mol(x, y)$ relation. Suppose, for example, we did not have any closure axioms in the axiomatization of cyclopropane (see Figure 6.15) – this would allow other rings containing other elements to satisfy the axiom (e.g., a ring full of sulfur atoms), which might not make chemical sense but it still satisfies the logical conditions for the axiom. We can see in Figure 6.15(b) that the closure axiom at the end enforces all atoms to be carbon atoms in the group, but in Figure 6.15(d), we can see that the not explicitly identifying the atoms allows for various possibilities of having different atoms that are not connected to be part of the group: carbon atoms c_4, c_5, and c_6 also satisfy the axioms in this description of cyclopropane, but because they are disconnected with the ring with c_1, c_2, and c_3, this is an unintended consequence of not having closure axioms.

With respect to the requirements presented in Chapter 3, we have fully satisfied Requirements (**R-1**), (**R-2**), and (**R-3**) since the elements of the domain are atoms, bonds, and functional groups, and have introduced notions of rings and chains to satisfy Requirement (**R-4**).

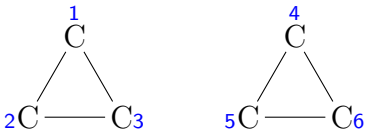
However, having these axioms that describe the components of the functional groups are *not* enough as they do not provide us with any insights on the functional groups. How can we represent the types of *connections* between one or more groups? With the axioms presented in this chapter, we are unable to distinguish between the various groups found within a compound that contains one or more groups. We discuss these connections in the next chapter.



(a)

$$\begin{aligned}\forall x\ cyclopropane(x) \equiv\ & ring(x) \wedge saturated(x) \wedge \\ & \exists c_1 \exists c_2 \exists c_3 \exists b_1 \exists b_2 \exists b_3 \\ & carbon(c_1) \wedge carbon(c_2) \wedge \\ & carbon(c_3) \wedge (c_1 \neq c_2) \wedge \\ & (c_1 \neq c_3) \wedge (c_2 \neq c_3) \wedge (b_1 \neq b_2) \wedge \\ & (b_1 \neq b_3) \wedge (b_2 \neq b_3) \wedge \\ & singlebond(b_1, c_1, c_2) \wedge \\ & singlebond(b_2, c_2, c_3) \wedge \\ & singlebond(b_3, c_3, c_1) \wedge \\ & (\forall y\ mol(y, x) \equiv \\ & ((y = c_1) \vee (y = c_2) \vee (y = c_3))\end{aligned}$$

(b) The proper axiomatization for cyclopropane in (a).



(c)

$$\begin{aligned}\forall x\ cyclopropane(x) \equiv\ & ring(x) \wedge saturated(x) \wedge \\ & \exists c_1 \exists c_2 \exists c_3 \exists b_1 \exists b_2 \exists b_3 \\ & carbon(c_1) \wedge carbon(c_2) \wedge \\ & carbon(c_3) \wedge (c_1 \neq c_2) \wedge \\ & (c_1 \neq c_3) \wedge (c_2 \neq c_3) \wedge \\ & (b_1 \neq b_2) \wedge (b_1 \neq b_3) \wedge \\ & (b_2 \neq b_3) \wedge \\ & singlebond(b_1, c_1, c_2) \wedge \\ & singlebond(b_2, c_2, c_3) \wedge \\ & singlebond(b_3, c_3, c_1)\end{aligned}$$

(d) Axiomatization without closure axiom. Lack of a closure axiom allows for various possibilities of atoms that satisfy the axiom in (c).

Figure 6.15: Axiomatizations of cyclopropane (C_2H_4), with and without closure axioms. In (b), the closure axiom outlined in blue ensures that c_1, c_2, and c_3 are the only atoms in the group. In (d), we have an instance where there are more carbon atoms c_4, c_5, and c_6 that are not connected with c_1, c_2 and c_3, but they still satisfy the definition of cyclopropane, since there are single bonds between c_4, c_5 and c_6. We can see that the lack of closure axioms may cause disconnected groups with similar atoms to satisfy the definitions of primitive functional groups.

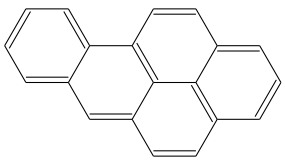
Chapter 7

Attaching Functional Groups Together

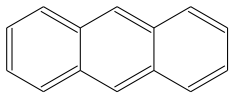
Before going into detail about the general attachments for functional groups, it would help to ease understanding by starting off with special classes of molecules and articulating the role the attachment constraints play in the ontology. We first start with a discussion of a highly-specific class of polycyclic molecules that satisfy assumptions of attachment, and then gradually weaken these assumptions as we progress through the chapter to generalize notions of attachment.

7.1 A Special Case: Purely Polycyclic Molecules

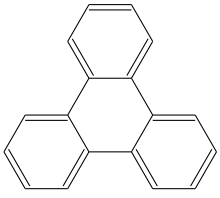
Consider purely polycyclic molecules: these are molecules where only rings exist and these rings are bonded together by fusion. An example of a purely polycyclic molecule is benzo[a]pyrene ($C_{20}H_{12}$), which is a polycyclic aromatic hydrocarbon (PAH), and the result of incomplete combustion of organic matter at temperatures; its structure is shown in Figure 7.1(a), where there are five benzene rings that are fused together, with no side chains. Other examples of polycyclic aromatic hydrocarbons include anthracene ($C_{14}H_{10}$) and triphenylene ($C_{18}H_{12}$), as shown in Figures 7.1(b) and 7.1(c), respectively. Anthracene consists of three fuzed benzene rings and triphenylene consists of four fused benzene rings; along with benzo(a)pyrene, they are compounds commonly found in coal tar.



(a) Benzo[a]pyrene ($C_{20}H_{12}$)



(b) Anthracene ($C_{14}H_{10}$)



(c) Triphenylene ($C_{18}H_{12}$)

Figure 7.1: Examples of polycyclic aromatic hydrocarbons.

Other polycyclic molecules include molecules belonging to the steroid family; the steroid core structure is composed of seventeen carbon atoms that are bonded together via four fused rings and is called gonane ($C_{17}H_{28}$), as shown in Figure 7.2. There are three six-membered cyclohexane rings and one five-membered cyclopentane rings that are fused together. Depending on which functional groups are attached to this core structure, the steroids vary in function.

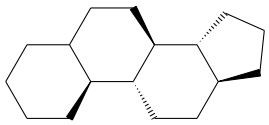


Figure 7.2: Gonane ($C_{17}H_{28}$), also known as perhydrocyclopenta[a]phenanthrene, is the steroid nucleus.

In the following sections, we first go over a special case of molecules that involves only molecules that contain rings and consider the following attachments between functional groups:

- Unique Fusion
- Unique Tethering
- Unique Spiro

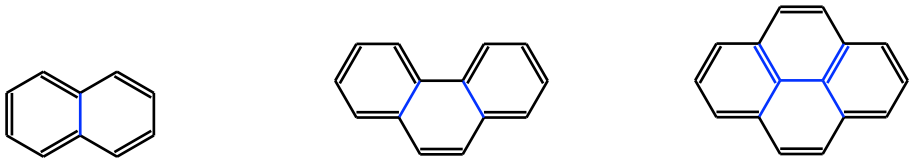
7.2 Unique Fusion Between Rings

From observing these polycyclic molecules, we notice that all groups in these molecules are rings, and that these rings are fused together. To represent this, we define a relation named $fused(g_1, g_2)$ where g_1 and g_2 are functional groups that are rings which are fused together. This fusion relationship is outlined in Definition 7.1 and formally defined in Axiom MAD-1, where a two atoms a_1 and a_2 are in both functional groups g_1 and g_2 with a bond b contained in both groups.

Definition 7.1: Fusion
Two groups are fused together if there is at least one bond shared between them.

$$\begin{aligned}\forall g_1 \forall g_2\,(fused(g_1,g_2) \equiv\ & (group(g_1) \wedge group(g_2) \wedge (g_1 \neq g_2) \wedge \\ & \exists a_1 \exists a_2 \exists b((atom(a_1) \wedge atom(a_2) \wedge \\ & (a_1 \neq a_2) \wedge mol(a_1,b) \wedge mol(a_2,b) \wedge \\ & mol(a_1,g_1) \wedge mol(a_1,g_2) \wedge \\ & mol(a_2,g_1) \wedge mol(a_2,g_2) \wedge \\ & mol(b,g_1) \wedge mol(b,g_2))))).\end{aligned} \tag{MAD-1}$$

Figure 7.3 outlines some examples of fused molecules – notice how we currently do not distinguish the differences between fusion with one bond and fusion with multiple bonds in these examples. Fused bonds are highlighted in blue in each of the three examples. For example, Figure 7.4 demonstrates a case of *unique* fusion, where only one bond is shared between the two cyclohexane rings in decalin.



(a) Naphthalene ($C_{10}H_6$) **(b)** Phenanthrene ($C_{14}H_{10}$) **(c)** Pyrene ($C_{16}H_{10}$)

Figure 7.3: Examples of fused molecules. The fused bonds are highlighted in blue. In (a), we have an example of a fused bond between two rings, and in (b), we have three fused rings that are fused along one bond. Notice how pyrene in (c) contains rings that are fused at multiple bonds.

However, not all molecules that exist are fused rings. We need to relax the constraints allow for different attachments of rings as Axiom MAD-1 is too strong to represent non-fused polycyclic molecules.

Polycyclic structures can also be linked via tethering or spiro connections, where there is a bond that connects two ring structures together, or an atom that connects two ring structures together, respectively. A classic example of *tethering* is biphenyl ($C_{12}H_{10}$), shown in Figure 7.5, which is an organic compound that forms colourless crystals and is mainly used as a heat transfer agent since it is stable and fairly non-reactive. Notice how there is a bond between the two phenyl groups in biphenyl: we can state that this bond is the tether between the two rings.

Likewise, another connection between two rings is the spiro connection, where two rings are connected by an atom that is shared between the rings. An example of a spiro connection between two ring structures would be spiro[5.5]undecane in Figure 7.6, which illustrates how two benzene rings are connected at one atom.

How do we represent these types of connections in the ontology? We must relax constraints in the ontology that all polycyclic molecules are fused together and allow the tethering connection, and then allow the spiro connection between rings. We discuss these in the sections that follow.

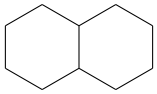


Figure 7.4: Decalin ($C_{10}H_{18}$)

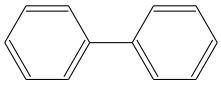


Figure 7.5: Biphenyl ($C_{12}H_{10}$)

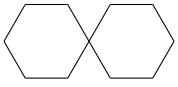


Figure 7.6: Spiro[5.5]undecane ($C_{11}H_{20}$)

7.3 Relaxing Constraints: Unique Tethering Between Rings

To allow this tethering connection between two rings, we introduce a ternary relation $tether(g_1, g_2, b)$ that takes three arguments: the first functional group, the second functional group, and the bond that connects them. We can define this tether relation as follows in Definition 7.2 and in Axiom MAD-2. With respect to Figure 7.5, we can state that the two benzene rings are *tethered* together. The tether bond is highlighted in blue in Figure 7.7.

Definition 7.2: Tether
Two functional groups are tethered together by a bond if there is a bond between the groups, where one atom in the bond is in the first functional group, and the other atom of the bond is in the second functional group.

$$\begin{aligned}\forall g_1 \forall g_2 \forall b\, (tether(g_1, g_2, b) \equiv\ & (group(g_1) \wedge group(g_2) \wedge \\ & bond(b) \wedge (g_1 \neq g_2) \wedge \exists a_1 \exists a_2 \\ & ((atom(a_1) \wedge atom(a_2) \wedge \\ & mol(a_1, g_1) \wedge mol(a_2, g_2) \wedge \\ & mol(a_1, b) \wedge mol(a_2, b) \wedge \\ & \neg mol(b, g_1) \wedge \neg mol(b, g_2))))).\end{aligned} \tag{MAD-2}$$

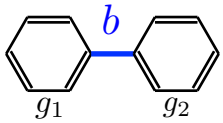


Figure 7.7: Example of a tether bond in biphenyl. The tether bond is highlighted in blue.

7.4 Relaxing Constraints: Unique Spiro Between Rings

With polycyclic molecules, the third type of connection between functional groups involves a shared atom between the rings. Spiro compounds present a 'twisted' structure of two or more rings, where two or three rings are linked together by one common atom. The common atom that connects the two (or sometimes three) rings is called the spiro atom. To represent this relationship, we define a new relation $spiro(g_1, g_2, a)$ which is a ternary relation that describes how two groups can be connected by one atom; we present this definition in Definition 7.3 and in Axiom MAD-3. The spiro atom is highlighted in blue in Figure 7.8.

Definition 7.3: Spiro
Two functional groups are connected together by spiro if they share one atom

between them, where said atom is in both groups.

$$\begin{aligned}\forall g_1 \forall g_2 \forall a\ (spiro(g_1, g_2, a) \equiv\ & (group(g_1) \wedge group(g_2) \wedge atom(a) \wedge \\ & (g_1 \neq g_2) \wedge \exists b_1 \exists b_2 ((bond(b_1) \wedge \\ & bond(b_2) \wedge mol(a, g_1) \wedge mol(a, g_2) \wedge \\ & mol(a, b_1) \wedge mol(a, b_2) \wedge mol(b_1, g_1) \wedge \\ & \neg mol(b_1, g_2) \wedge mol(b_2, g_2) \wedge \\ & \neg mol(b_2, g_1))))).\end{aligned} \tag{MAD-3}$$

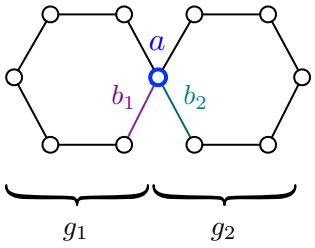


Figure 7.8: Example of a spiro atom, highlighted in blue.

Now that we have defined these two attachment relationships, we revisit the previous examples of molecules in Figures 7.5 and 7.6. We can write out axioms for both biphenyl and spiro[5.5]undecane; Example EX: biphenyl outlines how we define the biphenyl compound: there exist two benzene rings, g_1 and g_2, which are tethered together by a bond b.

$$\begin{aligned}\forall x\ biphenyl(x) \equiv\ & \exists g_1 \exists g_2 \exists b\ benzene(g_1) \wedge benzene(g_2) \wedge \\ & (g_1 \neq g_2) \wedge tether(g_1, g_2, b)\end{aligned} \tag{EX: biphenyl}$$

Similarly, Example EX: spiro55undecane shows how two cyclohexane rings are connected to each other via the spiro connection, and its corresponding axiomatization.

$$\begin{aligned}\forall x\ spiro55undecane(x) \equiv\ & \exists g_1 \exists g_2 \exists a\ cyclohexane(g_1) \wedge \\ & cyclohexane(g_2) \wedge (g_1 \neq g_2) \wedge \\ & spiro(g_1, g_2, a)\end{aligned} \tag{EX: spiro55undecane}$$

From these commonsense notions, consider a bond that is fused between two groups. In Axiom MA-1, we can state that two groups that are connected together via fusion have *at most* two atoms that shared between the groups, where these two atoms are in the same bond (that is also in both groups), and that these two groups are spiroed

between them; this is graphically depicted in Figure 7.9.

$$\begin{aligned}\forall a_1 \forall a_2 \forall a_3 \forall b \forall g_1 \forall g_2 \, ((&atom(a_1) \wedge atom(a_2) \wedge atom(a_3) \wedge group(g_1) \wedge \\ &group(g_2) \wedge (g_1 \neq g_2) \wedge bond(b) \wedge mol(b, g_1) \wedge \\ &mol(b, g_2) \wedge spiro(g_1, g_2, a_1) \wedge spiro(g_1, g_2, a_2) \wedge \\ &spiro(g_1, g_2, a_3)) \supset ((a_1 = a_2) \vee (a_1 = a_3) \vee \\ &(a_2 = a_3))).\end{aligned} \tag{MA-1}$$

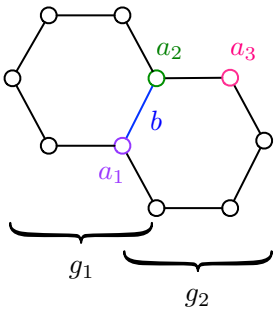


Figure 7.9: Visual depiction of Axiom MA-1.

7.5 Relaxing Constraints to Allow Chains & Rings

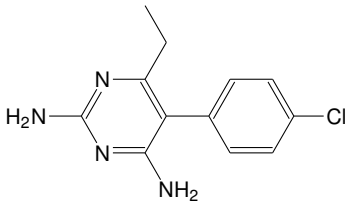
Now that we have the appropriate *attachment* relations for these ringed functional groups, we note that not all molecules in reality are polycyclic. What about functional groups that are chains? In order to allow chains to be described in molecules, we will need to relax the current restrictions on the ontology to allow functional groups to also be chains, not just rings.

7.5.1 Decomposing Chemical Graphs into Rings and Chains

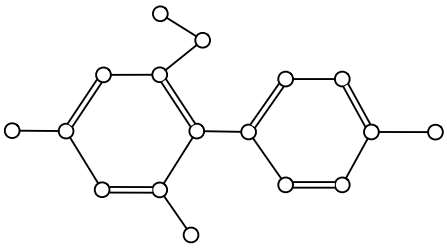
Before we begin to explain restrictions placed on how chains can be combined, we wish to bring up an important characteristic of the ontology:

> All chemical graphs can be decomposed into the attachment of primitive functional groups (rings and chains).

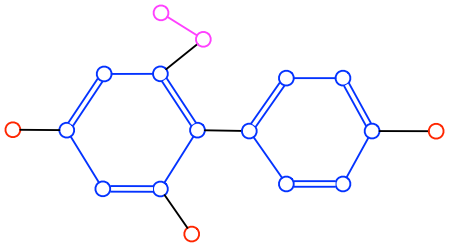
We discuss this decomposition theorem further in Chapter 14.5, but the main idea here is that, with a chemical graph, we can identify the rings and/or chains that compose the graph, along with any of the attachment relations between the various functional groups found in the graph. There is a 1 to 1 correspondence between the rings and chains identified in the graph with the primitive functional groups discussed in Chapter 6.3. For example, consider the pyramethamine molecule (trade name: Daraprim) in Figure 7.10(a). The underlying chemical graph is shown in Figure 7.10(b) and the decomposed version of the graph is shown in Figure 7.10(c), where the colours in figure denote the various primitive functional groups (rings and chains) that make up the graph.



(a) The Daraprim molecule.



(b) The underlying chemical graph.



(c) Components (rings and chains) of the molecule, where colour denotes different primitive functional groups.

Figure 7.10: Daraprim ($C_{12}H_{13}ClN_4$)

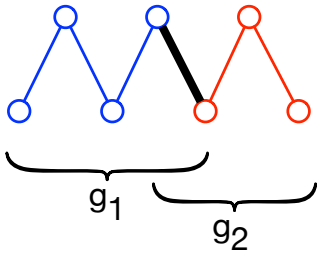
7.5.2 Constraining Chains

With chains, we do not allow the fusion attachment to occur between two chains, but allow two chains to be attached together via the spiro connection or the tether connection. The reason why we do not allow the fusion of two chains pertains to our *block decomposition* of the underlying graph and our usage of 2-connected components and paths of the graph – we point the reader to Chapter 14 for more information as it contains the mathematical rationales for designing the ontology in this manner. With respect to chains, we make the following *design decisions* for the ontology:

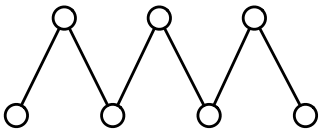
1. **Chains cannot be fused.** This is enforced due to the fact that it makes the decomposition of the underlying chemical graph too complex and may cause confusion when chained groups cannot be distinguished from one another. Figure 7.11(a) outlines an example of fused pentane and butane chains.

   The problem with allowing fused chains is that it may cause confusion when decomposing the chemical graph shown in Figure 7.11(b); if given the underlying graph without knowing which groups are fused together, it can result in the various decompositions seen in Figure 7.12. This poses a problem with describing the structure due to where the fused bond occurs – the possible combinations make it difficult to axiomatize the graph correctly.

2. **Chains *can* be connected via the spiro or tether attachments.** In the ontology, we permit the spiro and tether attachments for chains since they do not pose any complications when decomposing the underlying chemical graph into its core components: rings and chains. Figure 7.13 illustrates the spiro and tether connection between two chain groups.



(a) Fused pentane and butane chains. The pentane chain is labelled as g_1 and the butane chain labelled as g_2, with the fused bond indicated in bold.



(b) The underlying graph for this fused chain molecule.

Figure 7.11: Example of a fused chain and its underlying graph.

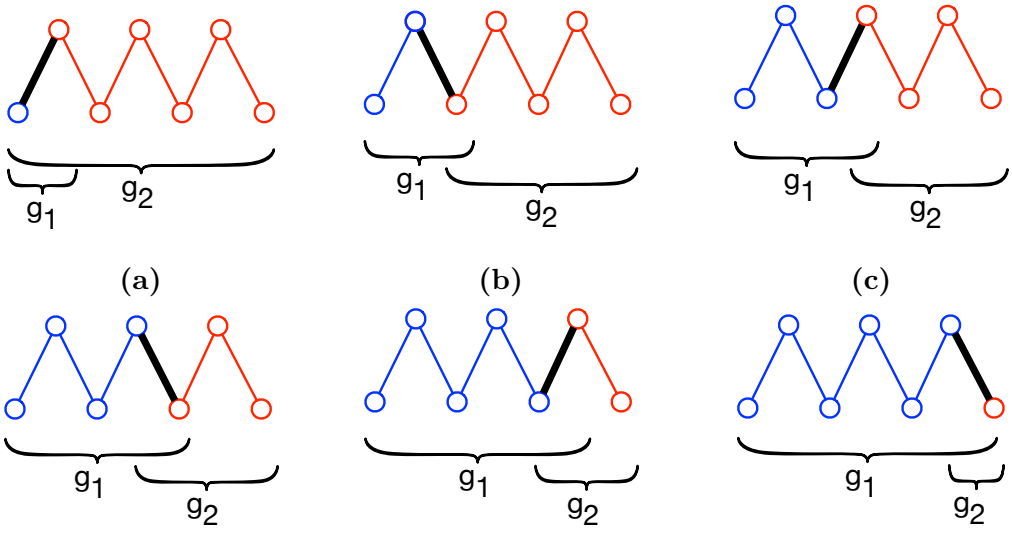


Figure 7.12: Examples of different ways of decomposing fused chains based on the underlying graph. The bolded bond is the fused bond, and the different functional groups are labelled as g_1 and g_2, respectively.

Figure 7.13: Examples how two chains g_1 and g_2 can be connected via the (a) spiro and (b) tethering connections. The bolded atom and bond in each scenario indicates the spiro atom and tether bond, respectively.

7.5.3 Generalizing Attachment Between Functional Groups

From these two restrictions on chains, we can make the following general statements that can be axiomatized:

- Axiom MA-2 states that, if two atoms, a_1 and a_2, are in the same functional group g, then any atom a_3 bonded between a_1 and a_2 is also in g.

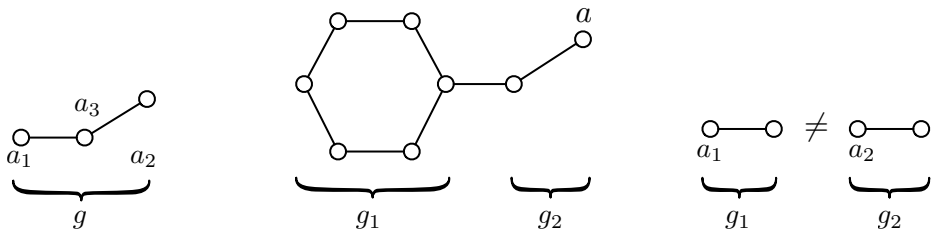
$$\begin{aligned}\forall a_1 \forall a_2 \forall a_3 \forall g_1 \forall g_2 \, ((&mol(a_1, g_1) \wedge mol(a_1, g_2) \wedge mol(a_2, g_1) \wedge \\ &mol(a_2, g_2) \wedge (a_1 \neq a_2) \wedge (g_1 \neq g_2) \wedge \\ &between(a_1, a_3, a_2)) \supset \\ &(mol(a_3, g_1) \wedge mol(a_3, g_2))).\end{aligned} \tag{MA-2}$$

- Axiom MA-3 states that, if there are two *distinct* functional groups, there exists an atom that is only in one of the groups and not the other.

$$\begin{aligned}\forall x \forall y \, ((&group(x) \wedge group(y) \wedge (x \neq y)) \supset \exists a((atom(a) \wedge \\ &mol(a, x) \wedge \neg mol(a, y)))).\end{aligned} \tag{MA-3}$$

- Axiom MA-4 states that functional groups cannot be coextensive. This means that two functional groups cannot have the exact same set of atoms. This axiom is required to differentiate between two groups of the same class.

$$\begin{aligned}\forall g_1 \forall g_2 \forall a_1 \forall a_2 \, ((&group(g_1) \wedge group(g_2) \wedge (g_1 \neq g_2) \wedge atom(a_1) \wedge \\ &atom(a_2) \wedge (a_1 \neq a_2) \wedge mol(a_1, g_1) \wedge mol(a_2, g_2)) \supset \\ &\neg mol(a_1, g_2) \wedge \neg mol(a_2, g_1))).\end{aligned} \tag{MA-4}$$



(a) Axiom MA-2 **(b)** Axiom MA-3 **(c)** Axiom MA-4

Figure 7.14: Visual depictions of Axioms MA-2, MA-3, and MA-4.

Now that we have gone over the different types of attachment between primitive functional groups, we can define a new relation $attached(g_1, g_2)$ in Definition MAD-4 which describes how two groups may be connected to each other via fusion, spiro, or tethering.

$$\begin{aligned}\forall g_1 \forall g_2 \, (attached(g_1, g_2) \equiv (&fused(g_1, g_2) \vee \\ &\exists a(spiro(g_1, g_2, a)) \vee \\ &\exists b(tether(g_1, g_2, b)))).\end{aligned} \tag{MAD-4}$$

7.6 Summary

In this chapter, we first presented a special case of molecules where all functional groups consist of rings, and then began to relax the constraints on how these polycyclic molecules can be attached together by introducing the notions of fusion, tether, and spiro, and discussed how these attachments are also used with chains. Then, we generalized these notions of attachment between functional groups. With respect to the requirements presented in Chapter 3, we have satisfied Requirements (**R-6**), (**R-7**), and (**R-8**) with the introduction of allowable attachments between functional groups.

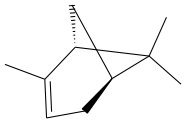


Figure 7.15: Pinene ($C_{10}H_{16}$)

However, these simple types of attachment are not representative of molecules that exist. What about molecules that appear to be what chemists call "bridged" (such pinene shown in Figure 7.15)? How does the ontology represent such molecules? What kinds of design decisions need to be made to ensure that bridges are represented in the ontology properly? In the next chapter, we discuss the advantages and disadvantages of these design decisions.

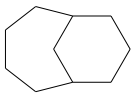
Chapter 8

Representing Bridges & Enforcing Attachment Restrictions

We have spent the previous chapter discussing the types of attachment between functional groups and only considering that each attachment type is unique:

- *Unique fusion*: groups are only allowed to fused at one bond
- *Unique spiro*: groups can only spiro at one atom, and
- *Unique tether*: groups can only have one tether

While we have taken these attachment requirements into consideration, we noticed that this is *not* reflective of the molecules that exist in the real world. There are various molecules that do not follow these uniqueness requirements, such as the bridged bicyclic molecules in Figure 8.1. For example, Figures 8.1(a) and 8.1(b) both have two rings that are fused at more than one bond.



(a) Bicyclo[4.3.1]decane ($C_{10}H_{18}$)

(b) Bicyclo[2.2.2]octane (C_8H_{14})

Figure 8.1: Examples of bridged bicyclic compounds.

Another example that is not as evident is with morphan, the core of the morphine molecule. Morphan is normally represented with a structure that has a bridged component in Figure 8.2(a). If we redraw the structure topologically, we get Figure 8.2(b) and can see that it appears to be multiple rings fused together across more than one bond, which violates the unique fusion condition introduced in the previous chapter.

How do we go about representing these kinds of molecules that violate unique fusion, unique spiro, and unique tethering while maintaining the 1:1 corresponding between rings and chains as cycles and paths in the underlying chemical graph?

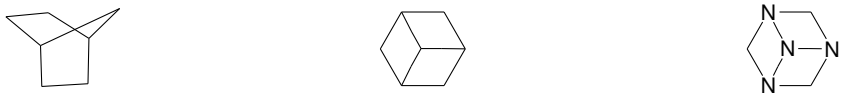
(a) Morphan structure **(b)** Morphan redrawn topologically

Figure 8.2: Morphan ($C_8H_{15}N$)

In this chapter, we discuss the notion of bridges in chemistry, and the design decisions made to axiomatize bridges in the ontology. We first provide an overview of bridge compounds and their components, how they violate the uniqueness of attachments discussed in Chapter 7, and the decisions that were made to accurately axiomatize such bridged molecules.

8.1 Bridged Compounds

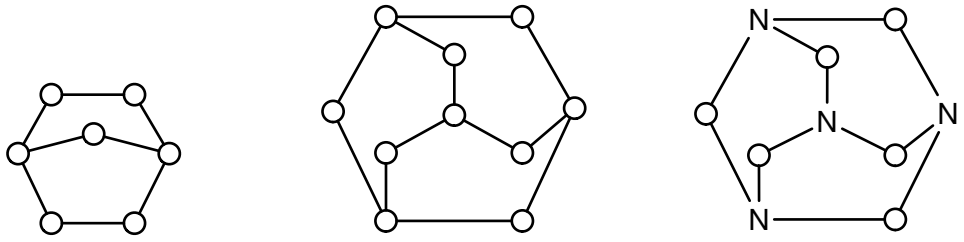
Bridges present an interesting representation challenge from a graph theory perspective, as they do not quite match up with the three unique attachment relations we presented earlier. Figure 8.3 outlines the norbornane, adamantane, and hexamine compounds that are classified to have bridges in their structures.



(a) Norbornane (C_7H_{12}) **(b)** Adamantane ($C_{10}H_{16}$) **(c)** Hexamine ($C_6H_{12}N_4$)

Figure 8.3: Examples of bridged compounds.

We can redraw these structure diagrams to show the topological graph of the structures. In Figure 8.4, we can see that each of the bridged structures in Figure 8.3 contain rings that appear to be fused together along more than one bond.



(a) Norbornane (C_7H_{12}) **(b)** Adamantane ($C_{10}H_{16}$) **(c)** Hexamine ($C_6H_{12}N_4$)

Figure 8.4: Examples of bridged compounds (redrawn topologically).

From these diagrams, we adopt the terminology used in naming complex polycyclic compounds from [Mos99] to describe components of bridged compounds. The author of [Mos99] provides an extension of the IUPAC nomenclature with additional guidance on how to name complex ring systems by introducing the following key terminology, which we will also use henceforth:

Definition 8.1: Bridged Compound (Adopted from [Mos99])
A bridged compound is a compound thats two or more rings that contain a bridge.

Definition 8.2: Bridgehead (Adopted from [Mos99])
A bridgehead atom is any atom that is part of the skeletal framework and is bonded to three or more skeletal atoms. Two bridgeheads are selected to be the main bridgeheads.

Definition 8.3: Bridge (Adopted from [Mos99])
A bridge is a single atom or an unbranched chain of atoms that connect two bridgeheads in a compound.

Definition 8.4: Main Ring (Adopted from [Mos99])
The main ring of a ring system is the ring that includes as many skeletal atoms of the polycylic compound as possible, and includes the two main bridgeheads.

Definition 8.5: Main Bridge (Adopted from [Mos99])
The main bridge is a bridge that connects two bridgeheads.

We present an abridged set of IUPAC rules for naming bridges in polycyclic compounds (abridged from [Mos99; Pur+93]) in Figure 8.5. Additional details on how to name modified ring systems that include heteroatoms, unsaturation, and stereochemistry can be found in [Mos99]. Other examples of bridged compounds are also shown in Figure 8.6.

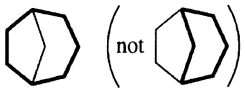
Now that we have introduced the relevant notions for bridges, we are now presented with the problem with how to represent bridges in compounds since it is not as straightforward as one would think.

8.2 Representing Bridges: Trade-Offs

Part of the issue with representing bridges is that it will affect how we recognize rings in the molecule. Bridges introduce an additional problem of having to count additional rings in the underlying graph – this is undesirable as additional rings may create problems when such rings are chemically infeasible in reality. In the previous chapter, we introduced the notion of polycyclic molecules that can be connected in three different ways: spiro,

IUPAC Bridge-naming Rules (Abridged from [Mos99; Pur+93])

1. A *main ring* is selected to include as many skeletal atoms of the polycyclic compound as possible. If there is more than one bridge, a main bridge is selected to include as many possible fo the atoms not in the main ring.

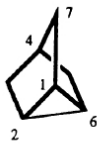


bicyclo[3.2.1]octane
not bicyclo[3.1.2]octane (7-membered main ring larger than 6)

2. A bicyclic system is named by:
   - The prefix 'bicyclo-',
   - Numbers indicating the bridge lengths, separated by full stops and placed in square brackets. Atoms of a bicyclic system are numbered from a bridgehead atom via the longest path to the second bridgehead atom, and bridge atoms are numbered from the lower numbered bridgehead atom.
   - The name of the hydrocarbon indicating the total number of skeletal atoms

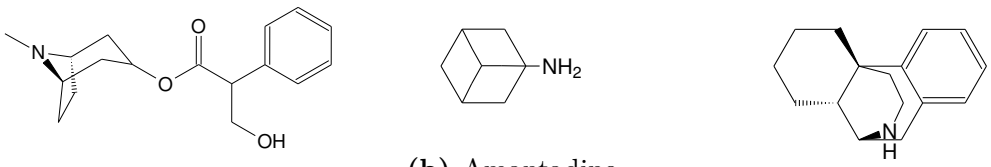
bicyclo[3.2.1]octane

3. For *polycyclic systems*, the number of atoms making up the secondary bridge are indicated in the compound name. The locants of the attachment points for each secondary bridge are cited as a pair of superscript locants separated by a comma. The name is then constructed by:
   - A prefix indicating the number of rings ('tricyclo-', 'tetracyclo-', etc.),
   - Numbers indicating the bridge lengths with the appropriate superscript locants, separated by full stops and placed in square brackets, and
   - The name of the parent hydrocarbon indicating the number of ring atoms.



tricyclo[2.2.1.$0^{2,6}$]heptane

Figure 8.5: Identifying bridges according to IUPAC naming rules (abridged from [Mos99; Pur+93]), with example figures from [Mos99].



(a) Atropine ($C_{17}H_{23}NO_3$) **(b)** Amantadine ($C_{10}H_{17}N$) **(c)** Morphinan ($C_{16}H_{21}N$)

Figure 8.6: Additional examples of bridged compounds.

tether, and fusion. However, the underlying assumption that each of these connections only occur at one unique atom (spiro) or one unique bond (tether, fusion) constrains our ability to represent rings. As we have seen in Figure 8.3, bridges violate these unique fusion constraints. What can we do to try to represent bridges? We are presented with the following options to represent bridges in the ontology:

Option #1: Maintain the unique attachment constraints, and represent bridges as an atom or chain that is connected via spiro at one end and tethered at the other end, or

Option #2: Relax the unique spiro constraint and allow the spiro attachment to occur at more than one atom between two functional groups, or

Option #3: Relax the unique tethering constraint and allow two functional groups to be tethered at more than two spots on the groups, or

Option #4: Relax the unique fusion constraint and allow fusion to occur along more than one bond between the two functional groups.

In the subsequent sections, we discuss the advantages and disadvantages of each of these four approaches, as they present interesting aspects that were not originally considered when only enforcing unique fusion, unique spiro, and unique tether.

8.2.1 Option: Bridges = An Atom or Chain Tethered at One End & Spiroed at the Other

What if we consider bridges to be elements in the domain of the ontology and that they are composed of two different types of attachment connections? Suppose bridges are composed of an atom or chain that is tethered at one end of the main ring and is connected via spiro at the other end of the same ring?

Figure 8.7 outlines what we mean by this composite connection for bridges. In this example, we have the topological graphs for morphan and norbornane, respectively. In both topological graphs, the colour in the diagram indicates the components that compose the bridge: one bridgehead is becomes the spiro atom (indicated in red), the chain or atom that is being connected (indicated in blue), and the second bridgehead is tethered to the chain in the ring (indicated in purple).

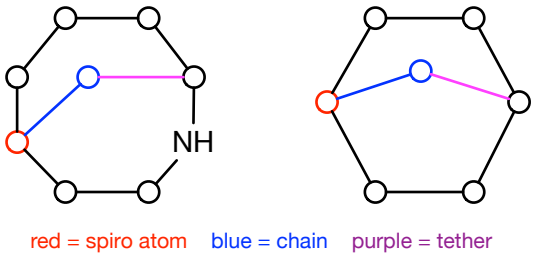


Figure 8.7: Bridges composed as an atom or a chain tethered at one end, and connected via spiro at the other.

However, this approach poses problems. Recall in the previous chapter that the intent of the ontology is to assign a correspondence between cycles in the underlying molecular graph with chemical rings. By allowing bridges to be represented in this manner, it allows for the potential confusion and creation of cycles that do not correspond to any rings (and therefore primitive functional groups that do not exist). For example, if we consider these topological graphs for morphan and norbornane, all of the cycles identified in Figure 8.8 might not correspond to primitive functional groups or chemically feasible rings.

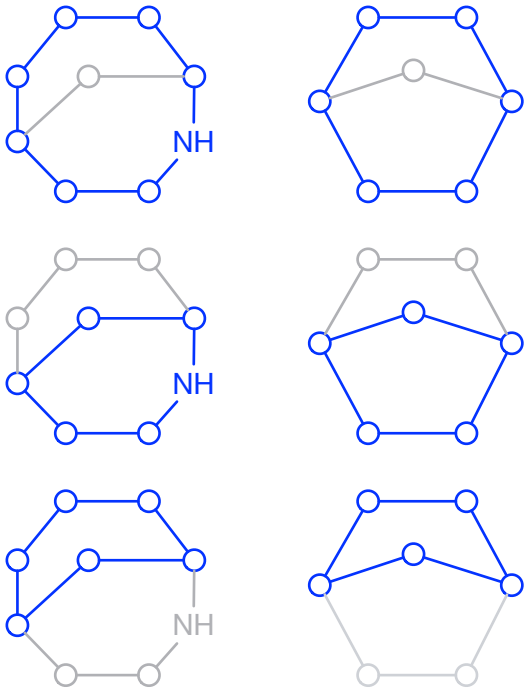


Figure 8.8: Cycles identified in the topological graphs for morphan ($C_8H_{15}N$) and norbornane (C_7H_{12}), respectively. Cycles are outlined in blue and the other atoms are intentionally greyed out.

Due to the issues with determining whether the cycles identified are valid chemical rings, this option of representing bridges poses unprecedented challenges for the cycle-ring correspondence and was removed from consideration.

8.2.2 Option: Allowing Multiple Spiro

The next alternative that was considered was to relax the unique spiro atom constraint to represent bridges. Figure 8.9 shows an example topological graph that could be allowed if we allow the spiro connection to occur on multiple atoms in a molecule. The bridge in this case would actually consist of two bridges from the two bridgeheads. The issue here becomes that there are very few molecules that have this kind of topological graph, and like before, there is the issue that there will no longer be a 1:1 correspondence between cycles identified in the underlying topological graph with primitive groups.

Figure 8.9: Relaxing spiro constraints to allow multiple spiro atoms. In (a), an example molecule with multiple spiro atoms is shown. In (b), the bolded atoms represent the spiro atoms, and the two functional groups are labelled as g_1 and g_2, respectively.

Allowing multiple spiro in an attempt to represent bridges poses similar issues presented in the previous approach: this creates multiple cycles in the topological graph that may or may not correspond to primitive functional groups. In Figure 8.10, we outline the various cycles that may be identified with this multiple spiro graph.

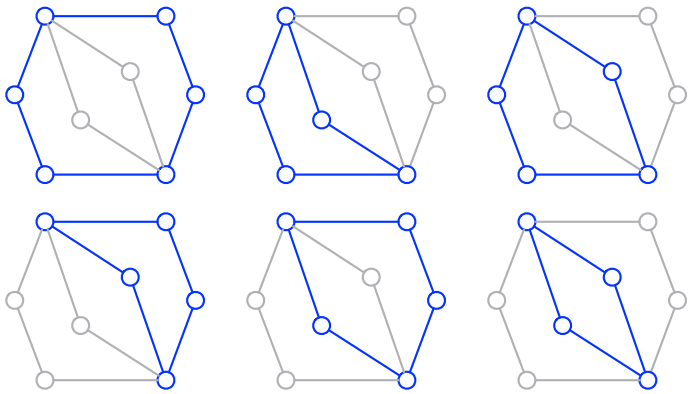


Figure 8.10: Cycles identified in the topological graph for this multiple spiro molecule. Cycles are outlined in blue and the other atoms are intentionally greyed out.

For graphs with multiple spiro atoms, this enumeration of cycles may become tedious and the various combinations may increase to unmanageable numbers to be mapped with an existing functional group. Consequently, we are presented with the same problem as before: the identified cycles do not correspond with functional groups that are rings.

8.2.3 Option: Allowing Multiple Tether

The next alternative to consider is to permit multiple tethers to occur between groups to represent bridges. Figure 8.11 outlines an example of what we mean by multiple tethering between two functional groups.

(a) (b)

Figure 8.11: Relaxing tether constraints to allow tether bonds. Here the bolded bonds represent the tether bonds, and the two functional groups are labelled as g_1 and g_2, respectively.

Like before, this poses the same problems as the first two options: additional cycles are created in the graph that may or may not correspond to primitive functional groups. In Figure 8.12, we outline the various cycles that may be identified with this multiple tether graph.

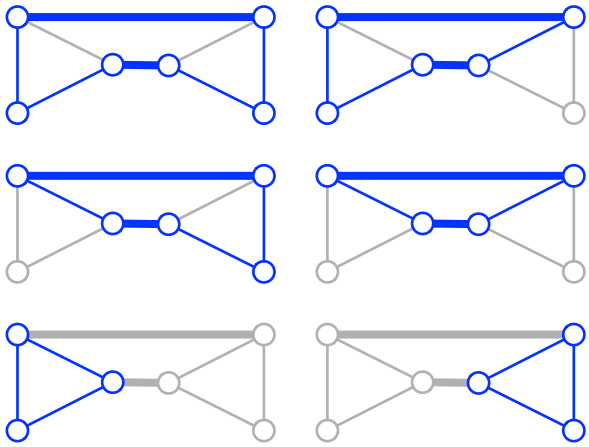
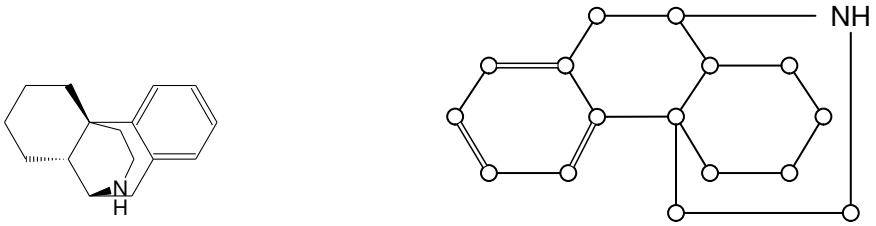


Figure 8.12: Cycles identified in the topological graph for this multiple tether molecule. Cycles are outlined in blue and the other atoms are intentionally greyed out.

8.2.4 Option: Allowing Multiple Fusions [Selected Option]

Consider morphan in Figure 8.2(a) and morphanin in Figure 8.13(a). Both molecules are similar in the sense there are multiple rings involved that appear to be fused together. Morphan in Figure 8.2(b) appears to have two bonds that are fused along the rings, and morphanin in Figure 8.13(b) appears to have six bonds that are fused together.

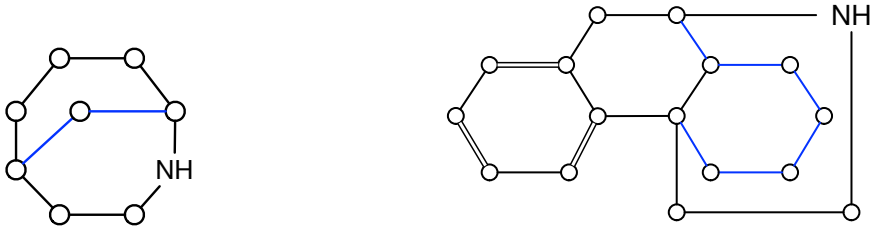
If we only allow unique fusion, how do we represent the bonds that appear to be fused between the rings (highlighted in blue in Figure 8.14)? The previous suggestions to allow multiple tethers, multiple spiro atoms, or even representing bridges as a composition of



(a) Morphinan structure **(b)** Morphinan redrawn topologically

Figure 8.13: Morphinan ($C_{16}H_{21}N$)

tether/spiro connections make it difficult to deal with when multiple bonds appear to be shared between more than two rings (as seen in Figure 8.14(b)).



(a) Morphan's fused bonds **(b)** Morphinan's fused bonds

Figure 8.14: Multiply-fused bonds in existing molecules, highlighted in blue.

Chemists also consider bicyclic (or even tricyclic) molecules to be composed of two or more rings, not as groups that are multiply-tethered or connected via multiple spiro atoms – this is evidenced through the von Baeyer system for naming polycyclic compounds (previously outlined Step #3 in Figure 8.5).

If we consider the alternative of representing bridges in the form of describing fused rings, it becomes a more feasible approach to handle bridges since chemists think of polycyclic molecules as fused rings and utilize name prefixes that indicate the number of rings being fused. If we consider *bridges* as a *relation* in the ontology, it permits us to describe bridges as a *property* of two (or more) fused rings. For example, consider varenicline in Figure 8.15 – this is a molecule that consists of rings and multiple bridges; its trade names are Chantix and Champix, and is primarily used to treat nicotine addiction.

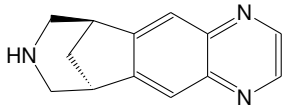


Figure 8.15: Varenicline (trade name: Chantix, $C_{13}H_{13}N_3$)

8.3 Design Decisions

From the discussion in the previous section, we have come up with the following ontological commitments and design decisions for bridges:

1. We only permit the spiro and tether attachments to be unique (i.e., groups can only spiro at one atom, and groups can be tethered by one bond). We enforce this with Axioms MBR-1 and MBR-2 in the ontology, as outlined below.

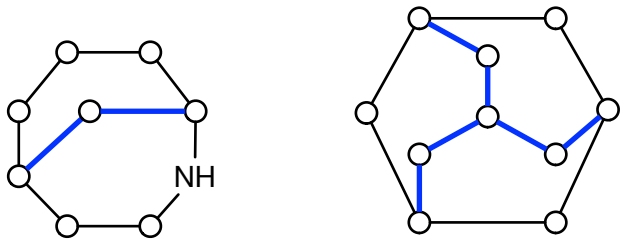
$$\exists a_1 \forall g_1 \forall g_2 \forall a_2 \left((spiro(g_1, g_2, a_1) \wedge spiro(g_1, g_2, a_2)) \supset (a_1 = a_2) \right). \qquad \text{(MBR-1)}$$

$$\exists b_1 \forall g_1 \forall g_2 \forall b_2 \left((tether(g_1, g_2, b_1) \wedge tether(g_1, g_2, b_2)) \supset (b_1 = b_2) \right). \qquad \text{(MBR-2)}$$

2. We permit multiple fusion in the ontology, where groups can be fused at multiple bonds. We do not constrain the fusion connection and allow the possibility of rings being fused on multiple bonds. We define a new relation, $multiply\_fused(g_1, g_2)$ in Defintion 8.6 and Axiom MBRD-1, which indicates two functional groups are fused at multiple bonds if and only if there are two (or more) bonds that are in both groups. Figure 8.16 illustrates the bonds (in bold) that are shared between the functional groups that make up morphan and adamantane, respectively.

Definition 8.6: Multiply Fused
Two functional groups are multiply fused together if and only if there are two or more bonds that are in both groups.

$$\begin{aligned} \forall g_1 \forall g_2 \, (multiply\_fused(g_1, g_2) \equiv\ & (group(g_1) \wedge group(g_2) \wedge (g_1 \neq g_2) \wedge \\ & \exists b_1 \exists b_2 (bond(b_1) \wedge bond(b_2) \wedge (b_1 \neq b_2) \wedge \\ & mol(b_1, g_1) \wedge mol(b_1, g_2) \wedge mol(b_2, g_1) \wedge \\ & mol(b_2, g_2)))). \end{aligned} \qquad \text{(MBRD-1)}$$



(a) Morphan ($C_8H_{15}N$) **(b)** Adamantane ($C_{10}H_{16}$)

Figure 8.16: Examples of multiply-fused compounds. Bonds that are bolded in blue indicate bonds that are found in two (or more) groups.

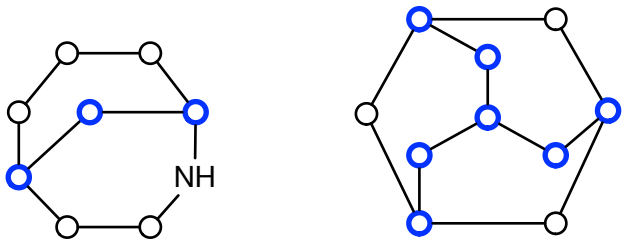
An additional thing to notice in these bridged molecules is that there are several atoms that are in multiple groups in a fused bond. We refer to these as *fused atoms* and define them in Definition 8.7 and in first-order as Axiom MBRD-2. This definition is graphically depicted in Figure 8.17. In Figure 8.18, these fused atoms found in morphan and adamantane are highlighted in blue.

Definition 8.7: Fused Atom
An atom in which it is in two functional groups that are fused together.

$$\begin{aligned}\forall a_1 \forall g_1 \forall g_2 \, (fusedAtom(a_1, g_1, g_2) \equiv (&atom(a_1) \wedge group(g_1) \wedge group(g_2) \wedge \\ &mol(a_1, g_1) \wedge mol(a_1, g_2) \wedge (g_1 \neq g_2) \wedge \\ &\exists b_1 \exists b_2 ((bond(b_1) \wedge bond(b_2) \wedge \\ &mol(a_1, b_1) \wedge mol(a_1, b_2) \wedge \\ &mol(b_1, g_1) \wedge mol(b_2, g_2))))).\end{aligned} \tag{MBRD-2}$$

(a) Unique fusion **(b)** Multiple fusion

Figure 8.17: Visual depictions of Axiom MBRD-2 with the unique fusion and multiply fused scenarios.



(a) Morphan ($C_8H_{15}N$) **(b)** Adamantane ($C_{10}H_{16}$)

Figure 8.18: Examples of fused atoms in morphan and adamantane. Atoms that are bolded in blue are fused atoms.

Furthermore, we can say that if two groups are fused together, then there exist *at least* two *distinct* fused atoms that belong in both groups (see Axiom MBR-3).

$$\begin{aligned}\forall g_1 \forall g_2 \, (fused(g_1, g_2) \supset \exists a_1 \exists a_2 ((&fusedAtom(a_1, g_1, g_2) \wedge \\ &fusedAtom(a_2, g_1, g_2) \wedge (a_1 \neq a_2)))).\end{aligned} \tag{MBR-3}$$

3. In the ontology, bridges are merely a label used to describe groups that are multiply fused together; the term 'bridge' is used to identify the atoms, bonds, and groups that are multiply-fused together. We do not necessarily consider bridges as elements in the ontology. Therefore, we consider $bridge(a, g_1, g_2)$ as a relation in the ontology to describe an atom a that is located in the bridge between two multiply-fused groups g_1 and g_2 (see Axiom MBRD-3).

$$\forall a \forall g_1 \forall g_2 \, (bridge(a, g_1, g_2) \equiv (atom(a) \wedge group(g_1) \wedge group(g_2) \wedge mol(a, g_1) \wedge mol(a, g_3) \wedge \forall g_3(((group(g_3) \wedge mol(a, g_3)) \supset (g_1 = g_3))) \wedge multiply\_fused(g_1, g_2))). \quad \text{(MBRD-3)}$$

Further, we also introduce the $bridgehead(a, g_1, g_2)$ relation in Axiom MBRD-4 to describe the atom in the skeletal ring that participates in bond with three or more non-hydrogen skeletal atoms.

$$\begin{aligned}\forall a \forall g_1 \forall g_2 \, (bridgehead(a, g_1, g_2) \equiv\ & (atom(a) \wedge group(g_1) \wedge group(g_2) \wedge \\ & (g_1 \neq g_2) \wedge \exists b \exists b_1 \exists b_2 ((bond(b) \wedge bond(b_1) \wedge \\ & bond(b_2) \wedge (b_1 \neq b_2) \wedge (b \neq b_1) \wedge (b \neq b_2) \wedge \\ & mol(a, b) \wedge mol(a, b_1) \wedge mol(a, b_2) \wedge \\ & mol(b, g_1) \wedge mol(b, g_2) \wedge mol(b_1, g_1) \wedge \\ & \neg mol(b_1, g_2) \wedge \neg mol(b_2, g_1) \wedge \\ & mol(b_2, g_2))))). \end{aligned} \quad \text{(MBRD-4)}$$

Additionally, there are at most two bridgeheads between two groups (see Axiom MBR-4).

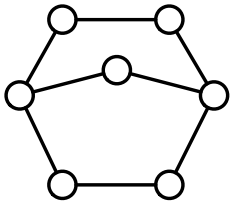
$$\begin{aligned}\forall a_1 \forall a_2 \forall a_3 \forall g_1 \forall g_2 \, ((&bridgehead(a_1, g_1, g_2) \wedge bridgehead(a_2, g_1, g_2) \wedge \\ &bridgehead(a_3, g_1, g_2)) \supset ((a_1 = a_2) \vee (a_1 = a_3) \vee \\ &(a_2 = a_3))). \end{aligned} \quad \text{(MBR-4)}$$

(a) Axiom MBRD-3 **(b)** Axiom MBRD-4

Figure 8.19: Visual depictions of Axiom MBRD-3 and MBRD-4.

8.4 Summary

With these bridge notions in mind, we can axiomatize the molecules that were presented at the beginning of the chapter. For example, norbornane in Figure 8.3(a) is often thought of by chemists as a pair of cyclopentane rings (C_5H_{10}) that share three of their five carbon atoms; it can be axiomatized as follows in Figure 8.20.



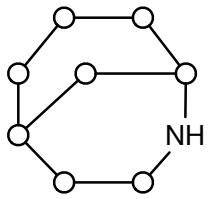
(a) Structure of norbornane

$$\forall x\, norbornane(x) \equiv \exists g_1 \exists g_2\, skeleton(x) \wedge cyclopentane(g_1) \wedge cyclopentane(g_2) \wedge multiply\_fused(g_1, g_2) \wedge mol(g_1, x) \wedge mol(g_2, x) \quad \text{(Sk:norbornane)}$$

(b) Axiom for norbornane

Figure 8.20: Axiomatizing norbornane.

We will discuss the $skeleton(x)$ relation in detail in Chapter 10, but for the purposes of showing this axiom for norbornane, we can think of it as a relation that specifies the skeletal framework of a molecule. Similarly, morphan in Figure 8.2(a), can be considered as a cyclohexane ring (C_6H_{12}) that is multiply-fused with a piperidine ring ($C_5H_{11}N$), and axiomatized as follows in Figure 8.21.



(a) Structure of morphan

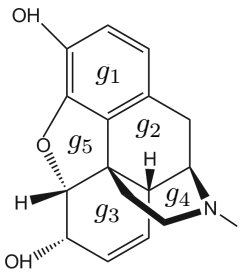
$$\forall x\, morphan(x) \equiv \exists g_1 \exists g_2\, skeleton(x) \wedge cyclohexane(g_1) \wedge piperidine(g_2) \wedge multiply\_fused(g_1, g_2) \wedge mol(g_1, x) \wedge mol(g_2, x) \quad \text{(Sk:morphan)}$$

(b) Axiom for morphan

Figure 8.21: Axiomatizing morphan.

Finally, with morphine, the compound consists of the following: phenolic ring (C_6H_6O), cyclohexane ring (C_6H_{12}), cyclohexenol ring (C_6H_8O), 1-methylpiperidine ring ($C_6H_{12}N$), and a partially saturated furan ring (C_4H_4O). We can axiomatize morphine as shown in Figure 8.22.

With respect to the requirements presented in Chapter 3, we have partially satisfied Requirement (**R-5**), as the axiomatization of bridges allows us to discuss how functional groups can be composed together. Now that we have committed to a set of design decisions and have axioms to describe bridged compounds, we will discuss how we distinguish between cycles found in a graph and known chemical rings.



(a) Structure of morphine

$$\begin{aligned}\forall x\, morphine(x) \equiv \exists g_1 \exists g_2 \exists g_3 \exists g_4 \exists g_5\, skeleton(x) \wedge phenol(g_1) \wedge \\ cyclohexane(g_2) \wedge cyclohexenol(g_3) \wedge \\ 1methylpiperidine(g_4) \wedge furan(g_5) \wedge \\ multiply\_fused(g_1, g_2) \wedge multiply\_fused(g_1, g_5) \wedge \\ multiply\_fused(g_2, g_3) \wedge multiply\_fused(g_2, g_4) \wedge \\ multiply\_fused(g_2, g_5) \wedge multiply\_fused(g_3, g_2) \wedge \\ multiply\_fused(g_3, g_4) \wedge multiply\_fused(g_3, g_5) \wedge \\ multiply\_fused(g_4, g_2) \wedge multiply\_fused(g_4, g_3) \wedge \\ mol(g_1, x) \wedge mol(g_2, x) \wedge \\ mol(g_3, x) \wedge mol(g_4, x) \wedge mol(g_5, x)\end{aligned} \tag{Sk:morphine}$$

(b) Axiom for morphine

Figure 8.22: Structure of morphine: g_1: phenolic ring, g_2: cyclohexane ring, g_3: cyclohexenol ring, g_4: 1-methylpiperidine ring, and g_4: partially saturated furan ring. (Adapted from Figure 2 in [BZÖ15])

Chapter 9

Orderings for Cycles & Rings

A problem when looking at the underlying graph of a molecule is that we do not know whether the cycles identified in the graph actually correspond to chemically-realizable rings. This normally would not be a problem but, since functional groups are elements in the domain of the ontology, we need to be able to identify whether or not cycles are indeed chemical rings. For example, in Figure 9.1, we do not know whether the cycles identified in blue are functional groups – are all cycles identified in a graph functional groups?

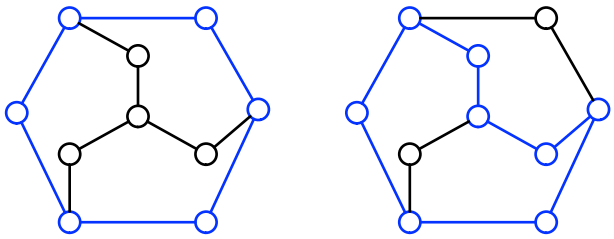


Figure 9.1: Topological graphs for morphan. Are the cycles outlined in blue chemical rings?

Examining the chemical graph boils down to the following questions using existing graph theory and mathematical notions:

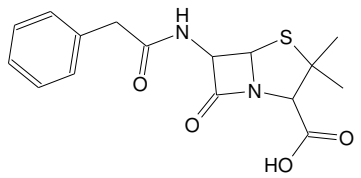
- How do we identify cycles in the underlying chemical graph? From those cycles, how do we identify these cycles a *simple cycles*?
- How do we associate these simple cycles as primitive functional groups that are chemical rings?
- What measures are taken to ensure disconnected cycles are the proper cycles we want?

We discuss how we answer these questions in the subsequent sections of this chapter.

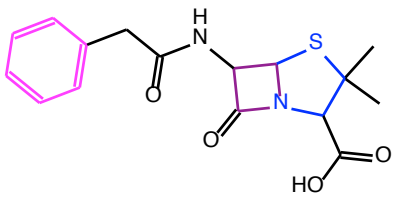
9.1 Finding Simple Cycles in the Graph

Since we have the underlying chemical graph of existing molecules, it is possible to find simple cycles in the graph via a breadth-first search algorithm. We consider simple

cycles as polygons in the chemical graph; for example in Figure 9.2(a), the penicillin molecule contains three simple cycles: the benzene ring (6-membered ring) and the beta β-lactam ring which consists of a fused 4-membered ring and a 5-membered ring. In shape terms, these are hexagonal, square, and pentagonal in shape, respectively. For this task of identifying simple cycles in chemical graphs, we treat the underlying graphs as *undirected* cyclic graphs.



(a) Penicillin ($C_9H_{11}N_2O_4S$)



(b) Simple cycles found in penicillin identified in magenta, dark purple, and blue, respectively.

Figure 9.2: Penicillin and its simple cycles.

Breadth-first search is used to traverse or search trees and graph structures, with an arbitrary node as the starting point and the algorithm explores neighbouring nodes of the graph before moving on to the next level of nodes. The algorithm explores all possible vertices from the supplied set of vertices (in this case, the adjacency matrices shown in Chapter 5.2). With this algorithm, the shortest path is returned – this mirrors the fact that simple cycles are the shortest paths in the cyclic graphs. The implementation of the algorithm used is discussed in greater detail in Chapter 14.7.2, however we would like to mention here that finding simple cycles in the graph only requires applying this algorithm to the chemical graph.

9.2 Associating Simple Cycles as Chemical Rings with Spanning Trees

In graph theory, we can consider acyclic graphs as graphs that do not contain any cycles – they are usually referred to as *forests*. Graphs that have cycles and/or connections are considered as *trees* (see Definition 9.1); Figure 9.3 illustrates examples of graphs that are considered trees.

We also consider the notion of a spanning tree in graph theory (see Definition 9.2): this is a subgraph which is a tree that includes all vertices of a given graph. Figure 9.4 outlines examples of graphs and its spanning trees.

Definition 9.1: Tree (Adopted from [Die12; DPV11])
A tree is an undirected graph that is connected and acyclic. Any connected, undirected graph $G = (V, E)$ *with* $|E| = |V| - 1$ *is a tree.*

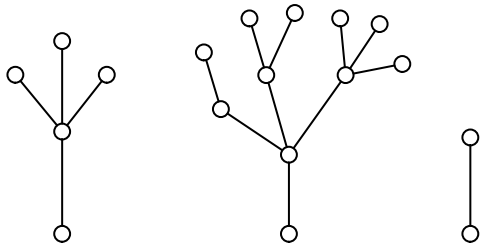


Figure 9.3: A forest of trees. The three graphs are considered trees, while the disjoint union of these trees are a forest.

Definition 9.2: Spanning Tree (Adopted from [Die12])
A spanning tree T of an undirected graph G is a subgraph that is a tree that includes all of the vertices of G. See Figure 9.4.

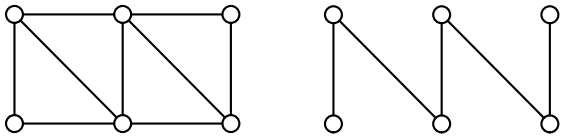


Figure 9.4: An example of a graph and its spanning tree.

Additionally, we introduce the notion of a fundamental cycle in Definition 9.3: this is a cycle that can be formed from a spanning tree of a graph. Adding one edge to a spanning tree will create a fundamental cycle – there is a distinct fundamental cycle for each edge that is not in the spanning tree.

Definition 9.3: Fundamental Cycle (Adopted from [Die12])
Consider a connected graph $G = (V, E)$ with a spanning tree $T \supseteq G$. For every edge $e \in E \setminus E(T)$, there is a unique cycle C_e in $T + e$: these cycles C_e are the fundamental cycles of G with respect to T. See Figure 9.5.

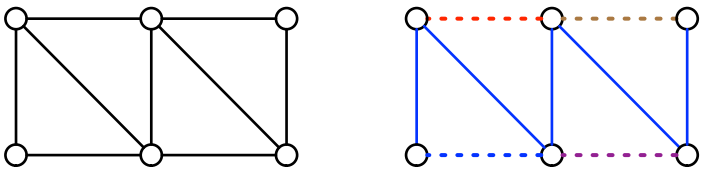


Figure 9.5: An example of a graph and its fundamental cycles. The dashed lines are edges in the original graph that form unique fundamental cycles, and the bolded blue edges represents the underlying spanning tree for the graph.

In Figure 9.5, the dashed lines are the edges in the original graph – each of these edges are in a unique fundamental cycle. As we will see later in this chapter with ringbonds,

we want to be able to axiomatize these edges in the ontology.

9.3 Cyclic Ordering

Now that we are able to identify the simple cycles in the graph, we also need to ensure that these cycles are cyclically ordered. The reason for this is that we do not want any disjoint cycles in the graph to constitute a functional group. For example, Figure 9.6 outlines two disconnected cycles that are not considered a functional group – however, up until this point, all the axioms that we have may actually consider these two disjoint cycles as a functional group in the underlying chemical graph.

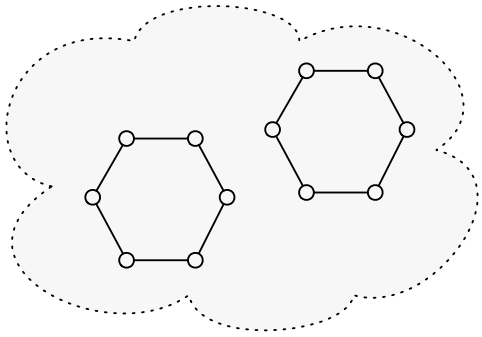


Figure 9.6: Two (or more) disconnected cycles should not be considered as *one* functional groups in the ontology.

We want to prevent these situations where the ontology may consider a primitive functional group to consist of two *disconnected* rings – in order to do this, we can leverage existing mathematical notions of *ordering*. In mathematics, a cyclic order is a way of arranging a set of objects in a circle, where a ternary relation is used [Hun16]; Figure 9.7 outlines an example of how atoms may be ordered in a ring.

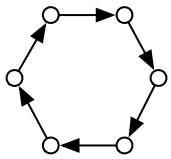


Figure 9.7: A cyclic ordering of atoms in a ring. Arrows denote ordering.

For molecular compounds that involve more than one ring, we need to be able to determine that one ring is different from the other; for example, Figure 9.8 outlines the orderings on the two benzene rings.

We introduce a new relation, $ringOrder(x, y, z)$, which enforces an order on x, y, and z, all of which are bonds in Axiom MCO-1.

$$\forall x \forall y \forall z \; (ringOrder(x, y, z) \supset (bond(x) \wedge bond(y) \wedge bond(z))). \qquad \text{(MCO-1)}$$

(a) Biphenyl ($C_{12}H_{10}$) **(b)** Ordering of rings in biphenyl.

Figure 9.8: Biphenyl graph and its orderings of atoms inside the rings. Arrows denote precedence.

Additionally, we introduce Axiom MCO-2 that states that if three bonds are ordered, there exists a ring where these bonds are located in.

$$\begin{aligned}\forall x \forall y \forall z\ ((ringOrder(x,y,z) \wedge bond(x) \wedge bond(y) \wedge bond(z)) \supset \\ \exists l((ring(l) \wedge mol(x,l) \wedge mol(y,l) \wedge mol(z,l)))).\end{aligned} \tag{MCO-2}$$

In Axiom MCO-3, the $ringOrder(x,y,z)$ relation is *asymmetric*.

$$\begin{aligned}\forall x \forall y \forall z \forall l\ ((ring(l) \wedge bond(x) \wedge bond(y) \wedge bond(z) \wedge \\ mol(x,l) \wedge mol(y,l) \wedge mol(z,l) \wedge \\ (x \neq y) \wedge (x \neq z) \wedge (z \neq y) \wedge \\ ringOrder(x,y,z)) \supset \neg ringOrder(z,y,x)).\end{aligned} \tag{MCO-3}$$

In Axiom MCO-4, bonds that are ordered in a ring are all *distinct* bonds.

$$\begin{aligned}\forall x \forall y \forall z \forall l\ ((ring(l) \wedge bond(x) \wedge bond(y) \wedge bond(z) \wedge \\ mol(x,l) \wedge mol(y,l) \wedge mol(z,l) \wedge \\ ringOrder(x,y,z)) \supset \\ (x \neq y) \wedge (x \neq z) \wedge (y \neq z))).\end{aligned} \tag{MCO-4}$$

In Axiom MCO-5, the $ringOrder(x,y,z)$ relation is cyclic: all distinct bonds in a ring have a bond that is ordered before or ordered after it.

$$\begin{aligned}\forall x \forall y \forall z \forall l\ ((ring(l) \wedge bond(x) \wedge bond(y) \wedge bond(z) \wedge \\ mol(x,l) \wedge mol(y,l) \wedge mol(z,l) \wedge \\ (x \neq y) \wedge (x \neq z) \wedge (z \neq y) \wedge \\ ringOrder(x,y,z)) \supset \forall w(((bond(w) \wedge mol(w,l)) \supset \\ (ringOrder(x,y,w) \vee ringOrder(w,y,z))))).\end{aligned} \tag{MCO-5}$$

In Axiom MCO-6, all rings have *at least* three bonds that are cyclically ordered.

$$\begin{aligned}\forall l((ring(l) \supset \exists x \exists y \exists z\ (bond(x) \wedge bond(y) \wedge bond(z) \wedge \\ ringOrder(x,y,z)))).\end{aligned} \tag{MCO-6}$$

In Axiom MCO-7, we permit the $ringOrder(x, y, z)$ relation to be transitive.

$$\begin{aligned}\forall x \forall y \forall z \forall w \forall l\ ((&ring(l) \wedge bond(x) \wedge bond(y) \wedge bond(z) \wedge bond(w) \wedge \\ &mol(x, l) \wedge mol(y, l) \wedge mol(z, l) \wedge mol(w, l) \wedge \\ &ringOrder(x, y, z) \wedge ringOrder(x, z, w)) \supset \\ &ringOrder(x, y, w)).\end{aligned} \tag{MCO-7}$$

With Axioms MCO-3, MCO-5, and MCO-6, $ringOrder(x, y, z)$ fulfills the asymmetric, cyclic, and transitive requirements for a *partial* cyclic order, respectively, as outlined in [Meg76].

9.4 Semilinear Betweenness of Rings

In addition to having a cyclic ordering on rings, we also want to be able to organize and refer to multiple rings in a molecule or skeleton. In polycyclic molecules, we would also want to be able to identify how one ring is between two others. For example, triphenyl (also known as P-Triphenyl) is a compound consisting of three rings – we can say that the second ring is *between* the other two in Figure 9.9.

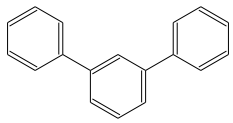


Figure 9.9: P-Triphenyl ($C_{18}H_{14}$)

We adopt the existing notion of betweenness in mathematics with Definition 9.4.

> **Definition 9.4: Semilinear Betweenness (Adopted from [Grü11])**
> *A semilinear betweenness relation B is a ternary relation that is definable in a semilinear ordering $\left\langle X, < \right\rangle$ by the formula:*
>
> $$\forall x \forall y \forall z\ B(x, y, z) \equiv (((x < y) \wedge (y < z)) \vee ((z < y) \wedge (y < x)))$$

With this in mind, we can state the following sentences in the ontology by using the $betweenMol(x, y, z)$ relation to be our betweenness relation:

- We enforce semilinear betweenness on the atoms found in a ring in Axiom MBTWN-1: atoms are between themselves. This means that we can state that an atom y is between atoms x and z.

$$\forall x \forall y \forall z\ (betweenMol(x, y, z) \supset (atom(x) \wedge atom(y) \wedge atom(z))). \tag{MBTWN-1}$$

- In Axiom MBTWN-2, the betweenness relation is symmetric – an atom y is between

x and z.

$$\forall x \forall y \forall z \forall s \, ((skeleton(s) \wedge atom(x) \wedge atom(y) \wedge atom(z) \wedge mol(x,s) \wedge mol(y,s) \wedge mol(z,s) \wedge betweenMol(x,y,z)) \supset betweenMol(z,y,x)). \quad \text{(MBTWN-2)}$$

- In Axiom MBTWN-3, atoms in a skeleton are between themselves.

$$\forall x \forall y \forall z \forall s \, ((skeleton(s) \wedge atom(x) \wedge atom(y) \wedge atom(z) \wedge mol(x,s) \wedge mol(y,s) \wedge mol(z,s) \wedge betweenMol(y,x,z) \wedge betweenMol(x,y,z)) \supset (x = y)). \quad \text{(MBTWN-3)}$$

- In Axiom MBTWN-4, at most two atoms are between each other in a skeleton.

$$\forall x \forall y \forall z \forall w \forall s \, ((skeleton(s) \wedge atom(x) \wedge atom(y) \wedge atom(z) \wedge mol(x,s) \wedge mol(y,s) \wedge mol(z,s) \wedge mol(w,s) \wedge betweenMol(y,x,z)) \supset (betweenMol(y,x,w) \vee betweenMol(z,x,w))). \quad \text{(MBTWN-4)}$$

- In Axiom MBTWN-5, a point between two bridgeheads is also part of both groups that are fused together.

$$\forall a_1 \forall a_2 \forall a_3 \forall g_1 \forall g_2 \, ((mol(a_1,g_1) \wedge mol(a_1,g_2) \wedge mol(a_2,g_1) \wedge mol(a_2,g_2) \wedge (a_1 \neq a_2) \wedge (g_1 \neq g_2) \wedge betweenMol(a_1,a_3,a_2)) \supset (mol(a_3,g_1) \wedge mol(a_3,g_2))). \quad \text{(MBTWN-5)}$$

- In Axiom MBTWN-6, all skeletons are connected due to semilinear betweenness: this means that skeletons are either in same bond or there is an atom between them.

$$\forall x \forall y \forall s \, (skeleton(s) \wedge atom(x) \wedge atom(y) \wedge mol(x,s) \wedge mol(y,s) \supset (\exists z(atom(z) \wedge betweenMol(x,z,y))) \vee (\exists b(mol(x,b) \wedge mol(y,b) \wedge bond(b)))). \quad \text{(MBTWN-6)}$$

For Axioms MBTWN-2, MBTWN-3, MBTWN-4, and MBTWN-6, the $skeleton(x)$ relation is used – we discuss this further in Chapter 10, but the notion of betweenness still applies for atoms in functional groups and in skeletons.

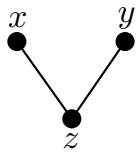
9.5 Ring Bonds

With the notions of spanning trees and fundamental cycles, we introduce the concept of a *ringbond*, which is essentially a bond that is found inside a ring with vertices that are *incomparable*, as defined in Definition 9.5 and in Axiom MRBD-1. Recall that fundamental cycles can be formed from a spanning tree of a graph: in the context of MoSt, these are the bonds of a chemical graph that belong in rings ('ringbonds').

Definition 9.5: Ringbond
A ringbond is a bond in a ring but whose vertices are not comparable by the $betweenMol(x, y)$ *relation.*

$$\begin{aligned}\forall b\, ringBond(b) \equiv (&bond(b) \wedge \exists a_1 \exists a_2((atom(a_1) \wedge atom(a_2) \wedge \\ &mol(a_1, b) \wedge mol(a_2, b) \wedge \neg(\exists a_3((atom(a_3) \wedge \\ &(betweenMol(a_3, a_1, a_2) \vee \\ &betweenMol(a_1, a_2, a_3))))))).\end{aligned} \tag{MRBD-1}$$

We make the distinction here that two elements are *incomparable* in the context of partially ordered sets in mathematics. Any two elements x and y of a set P that is partially ordered by a binary relation $\leq$ are comparable when either $x \leq y$ or $y \leq x$. If it is not the case that x and y are comparable, then they are called *incomparable*. We show this graphically in Figure 9.10, where we have three points x, y, and z, where $B(x, z, y)$ denotes that z is between x and y. The $incomparable(x, y)$ relation enforces the fact that x are y are not between any other points. In the context of MoSt, we utilize the semilinear betweenness relation, $betweenMol(x, y, z)$, to handle betweenness of atoms in a skeleton.



$B(x, z, y)$ $\quad \forall x \forall y\, incomparable(x, y) \equiv \neg(\exists z\, B(x, y, z) \vee B(y, x, z))$

Figure 9.10: Incomparable vertices x and y.

Using this notion of ringbond, we can also axiomatize this notion that, if a bond is a ringbond, then it must be in a ring in Axiom MRB-1

$$\forall b\, (ringBond(b) \supset \exists g((mol(b, g) \wedge ring(g)))). \tag{MRB-1}$$

For example, in Figure 9.11, the bond highlighted in blue is an example of a ringbond (and thus every bond in this benzene ring is a ringbond).

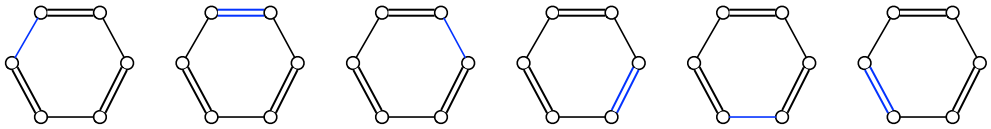


Figure 9.11: Example of ringbonds in a benzene ring are highlighted in blue. (All bonds in this benzene ring are ringbonds.)

9.6 Summary

In this chapter, we have discussed the motivations for interweaving the ontology with notions from graph theory to ensure cycles identified in chemical graphs correspond to chemical rings. With respect to the requirements presented in Chapter 3, we have partially satisfied Requirement (**R-5**). Furthermore, we have shown how cyclic ordering and semilinear betweenness are important concepts that needed to be added to the ontology to ensure that we axiomatize *connected* molecules.

Chapter 10

Skeletons: Putting It All Together

Now that we have all of the necessary pieces of the ontology, we can put them together. Recall that a molecular graph contains nodes that represent non-hydrogen atoms and the covalent bonds between them (Definition 5.2). From the list of ontological commitments presented in Chapter 3.4, Requirement (**R-5**) indicates that we must be able to represent the skeleton of a molecule. In this chapter, we discuss the notion of a skeleton and now we can axiomatize properties of skeletons with the ontology.

10.1 Esters: Different Ways of Looking At Molecules

With the properties of skeletons in mind, we can demonstrate the power of the ontology: we are able to compose molecules in various ways. A real-world example would be with esters in chemistry.

Esters are compounds derived from acids, in which at least one hydroxyl group ($-OH$) is replaced by an alkoxy group($-O-alkyl$) – these are usually derived from a carboxylic acid and an alcohol [IUP06]. Figure 10.1 below shows the general structure of esters, where the R can denote a hydrogen atom, an alkyl group, or an aryl group, and R' can denote an alkyl group or aryl group.

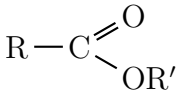
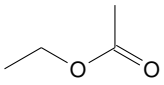


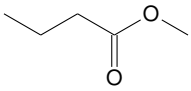
Figure 10.1: General structure of esters.

Esters feature a carbon-to-oxygen double bond that is singly bonded to a second hydrogen atom, which is then joined to an alkyl or an aryl group. Figure 10.2 shows ethyl acetate and methyl butyrate. Esters that are derived from simple carboxylic acids are named by their common names, as opposed to their IUPAC names (shown in Table 10.1).

One thing we noticed while examining esters was how they can be partitioned. For example, ethyl acetate in Figure 10.2(a) is derived from an alcohol, and an acyl group derived from a carboxylic acid. We reproduce the skeletal diagram in Figure 10.3 with



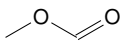
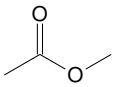
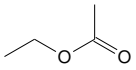
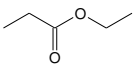
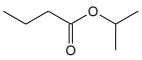
(a) Ethyl Acetate ($C_4H_8O_2$)



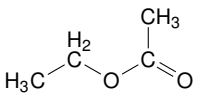
(b) Methyl Butyrate ($C_5H_{10}O_2$)

Figure 10.2: Examples of esters.

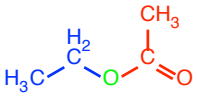
Table 10.1: Example of common and IUPAC names for esters.

Structural Formula	Skeletal Formula	Common Name	IUPAC Name
$HCOOCH_3$		methyl formate	methyl methanoate
CH_3COOCH_3		methyl acetate	methyl ethanoate
$CH_3COOCH_2CH_3$		ethyl acetate	ethyl ethanoate
$CH_3CH_2COOCH_2CH_3$		ethyl propionate	ethyl propanoate
$CH_3CH_2CH_2COOCH(CH_3)_2$		isopropyl butyrate	isopropyl butanoate

hydrogens included in the diagram, and outline the alcohol group in blue and the acyl group in red, and indicate that these groups overlap at the oxygen atom in green.



(a) Skeletal diagram with hydrogens

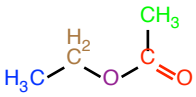


(b) Composition of alcohol (blue) and acyl group (red). Both groups overlap at the oxygen atom (green).

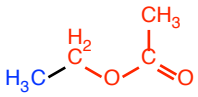
Figure 10.3: Ethyl acetate and its composition.

Looking at Figure 10.3(b), we can also partition the molecule into its simple functional groups, or as a composition of composite groups. Alternatively, we can look at its decomposition in terms of a composition of primitive functional groups in Figure 10.4(a), or as the composition of a composite functional group with a primitive group in Figure 10.4(b). Various combinations are possible.

How can we represent these various ways of dividing up the molecule? We need to



(a) Decomposition with all primitive functional groups.



(b) Decomposition resulting in methyl acetate and a methyl group.

Figure 10.4: Various ways of decomposing ethyl acetate. Colours denote groups.

introduce the concept of *skeletons* in the ontology to allow us to partition a molecule in various ways.

10.2 Skeletons & Their Properties

Recall that a skeletal formula serves as a shorthand representation of the bonding and geometry of molecules. In addition to this, it shows the skeletal structure (or *skeleton*) of a molecule.

In the ontology, we introduce a new class, $skeleton(x)$, as an element of the domain: this relation signifies a portion or the entirety of a molecule. Skeletons can be composed of other skeletons – they allow us to partition the structure of molecules into various pieces, along with also combining pieces together.

> **Definition 10.1: Skeleton**
> *A skeleton in MoSt is the composition of one or more functional groups that are attached together. A skeleton can be composed of several skeletons.*

For example, consider ethyl acetate: we can write axioms for this molecule using the decompositions presented in Figure 10.5 in various ways. First, consider the bottom half of the figure where the functional groups are outlined in different colours: g_1 corresponds to a methyl group (in red), g_2 corresponds to a carbonyl group (in green), g_3 corresponds to an oxygen group (in purple), and g_4 corresponds to an ethyl group (in blue). These four groups are connected via the $mol(x, y)$ relation in the ontology, outlined with the solid black line between the groups in the figure.

Further, the blue dotted lines in Figure 10.5 shows how the primitive functional groups can compose the various skeletons – recall that we define the notion of a skeleton to contain one *or more* functional groups; this means that we can have a skeleton that is composed of one functional group, as well as another skeleton that is composed of multiple functional groups. In Figure 10.5, we have the following:

- Skeleton s_2 consists of groups g_3 and g_4,
- Skeleton s_3 is composed of skeleton s_1 and the skeleton for group g_3,
- Skeleton s_1 consists of the skeletons for groups g_1 and g_2, and
- Skeleton s_4 contains skeletons s_1, s_2, and s_3 as parts.

The compositions of these various skeletons are shown to the right of the figure, with the colours indicating which groups compose which skeletons. We can axiomatize these four

different approaches to breaking down ethyl acetate:

1. In (Sk:ethyl acetate-1), skeleton s_4 is the skeleton for the entire ethyl acetate compound (s_4 is relabelled as x in the axiom):

$$\forall x\, ethyl\_acetate(x) \supset\ skeleton(x) \qquad \text{(Sk:ethyl acetate-1)}$$

2. In (Sk:ethyl acetate-2), ethyl acetate can be defined as the composition of two skeletons for acetic acid (s_1) and ethanol (s_2) that are tethered together:

$$\begin{aligned}\forall x\, ethyl\_acetate(x) \equiv \exists s_1 \exists s_2 \exists b_1\ & skeleton(x) \wedge \\ & acetic\_acid(s_1) \wedge ethanol(s_2) \wedge \\ & mol(s_1, x) \wedge mol(s_2, x) \wedge \\ & tether(s_1, s_2, b_1)\end{aligned} \qquad \text{(Sk:ethyl acetate-2)}$$

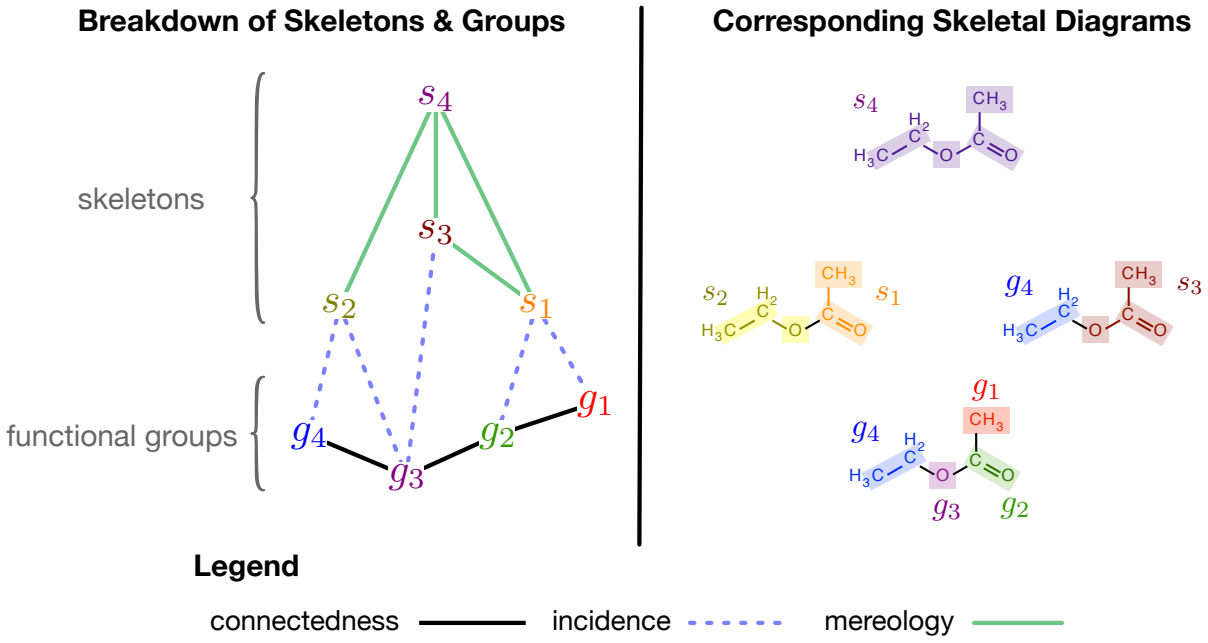


Figure 10.5: Composition of the skeleton for ethyl acetate. g_1, g_2, g_3, and g_4 signify the primitive functional groups, and s_1, s_2, s_3, and s_4 signify the skeletons, respectively. The primitive functional groups are connected via the $mol(x, y)$ relation using the dark, bolded black lines. Skeletons that contain functional groups are outlined in the dotted blue lines in the figure; for example, s_2 consists of g_3 and g_4. Green lines show parthood between skeletons.

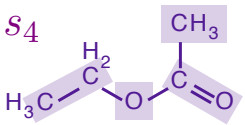


Figure 10.6: Graphical depiction of the composition of (Sk:ethyl acetate-1).

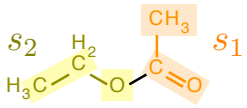


Figure 10.7: Graphical depiction of the composition of (Sk:ethyl acetate-2).

3. In (Sk:ethyl acetate-3), ethyl acetate can be defined as the tethered composition of the acetyl-oxy skeleton (s_3) and the skeleton for the ethane group (g_4):

$$\begin{aligned}\forall x\ ethyl\_acetate(x) \equiv \exists g_4 \exists s_3 \exists b_1\ & skeleton(x) \wedge \\ & acetyl\_oxy(s_3) \wedge \\ & ethane(g_4) \wedge mol(s_3, x) \wedge \\ & mol(g_4, x) \wedge tether(s_1, s_2, b_1)\end{aligned} \qquad \text{(Sk:ethyl acetate-3)}$$

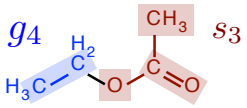


Figure 10.8: Graphical depiction of the composition of (Sk:ethyl acetate-3).

4. In (Sk:ethyl acetate-4), ethyl acetate can be defined as the composition of the skeletons of four primitive groups: methyl (g_1), carboxyl (g_2), ether (g_3), and ethane (g_4).

$$\begin{aligned}\forall x\ ethyl\_acetate(x) \equiv \exists g_1 \exists g_2 \exists g_3 \exists g_4 \exists b_1 \exists b_2 \exists b_3\ & skeleton(x) \wedge \\ & methyl(g_1) \wedge carbonyl(g_2) \wedge \\ & ether(g_3) \wedge ethane(g_4) \wedge \\ & mol(g_1, x) \wedge mol(g_2, x) \wedge \\ & mol(g_3, x) \wedge mol(g_4, x) \wedge \\ & tether(g_1, g_2, b_1) \wedge tether(g_2, g_3, b_2) \wedge \\ & tether(g_3, g_4, b_3)\end{aligned} \qquad \text{(Sk:ethyl acetate-4)}$$

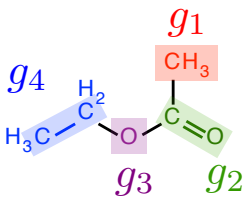


Figure 10.9: Graphical depiction of the composition of (Sk:ethyl acetate-4).

We can see that the notion of a skeleton allows for flexibility in how one can axiomatize compounds that consist of more than one functional group. These decompositions of a compound resemble what we call *attachment graphs* in Chapter 14.7.1 (and in Definition 14.8).

From these approaches, we can see that skeletons allow for versatility in the axiomatization of small molecules with the ontology. We are free to choose which pieces (primitive functional groups or skeletons) can be used to write out the axioms.

We also note here that this new $skeleton(x)$ relation is disjoint from the $atom(x)$, $bond(x)$, and $group(x)$ relations, as indicated in Axiom MS-1.

$$\forall x\,(skeleton(x) \supset \neg((atom(x) \vee bond(x) \vee group(x)))). \tag{MS-1}$$

With this new relation, we can axiomatize the properties of skeletons as follows:

- In Axiom MS-2, we state that skeletons are not incident with one another – we will be utilizing a mereology in Axiom MSD-3 to illustrate that skeletons can be a *part* of each other.

$$\forall x \forall y\,((mol(x,y) \wedge skeleton(x) \wedge skeleton(y)) \supset (x = y)). \tag{MS-2}$$

- In Axiom MS-3, we state that every skeleton has a primitive functional group. As skeletons represent a portion or the entirety of a molecule, we enforce this notion that a skeleton must contain at least one functional group.

$$\forall s\,(skeleton(s) \supset \exists g((group(g) \wedge mol(g,s)))). \tag{MS-3}$$

- Additionally, in Axiom MS-4, we identify that each primitive functional group in a skeleton contains an atom that is in a bond with an atom in a different functional group in the skeleton. This is trivially satisfied if the skeleton only contains one group.

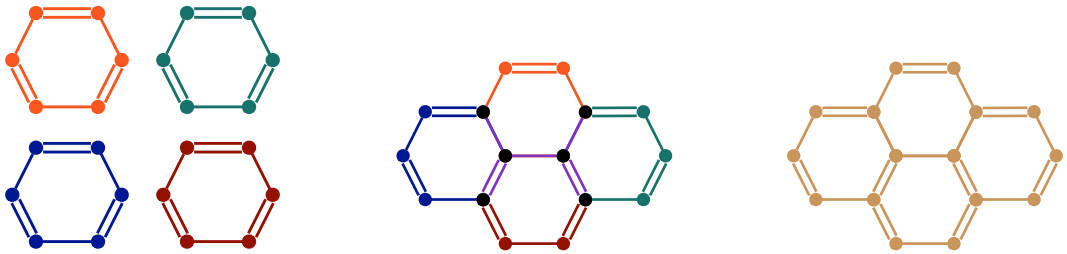
$$\begin{aligned}\forall g_1 \forall s\,((&skeleton(s) \wedge group(g_1) \wedge mol(g_1,s)) \supset \\ &\exists a_1 \exists a_2 \exists b \exists g_2((atom(a_1) \wedge atom(a_2) \wedge bond(b) \wedge \\ &group(g_2) \wedge (g_1 \neq g_2) \wedge mol(a_1,g_1) \wedge mol(a_2,g_2) \wedge \\ &mol(a_1,b) \wedge mol(a_2,b)))).\end{aligned} \tag{MS-4}$$

- In Axiom MSD-1, we introduce the definition of fused skeletons that only contain fused rings in the form of definition.

$$\begin{aligned}\forall s\,(fusedSkeleton(s) \equiv (&skeleton(s) \wedge \forall g_1 \forall g_2(((group(g_1) \wedge mol(g_1,s) \wedge \\ &group(g_2) \wedge mol(g_2,s) \wedge (g_1 \neq g_2)) \supset \\ &(ring(g_1) \wedge ring(g_2) \wedge fused(g_1,g_2)))))).\end{aligned} \tag{MSD-1}$$

For example, consider the four benzene rings (C_6H_6) outlined in different colours in

Figure 10.10; we can fuse these rings together to form a pyrene ($C_{16}H_{10}$) skeleton, which we can consider as fused skeleton.



(a) **(b)** **(c)**

Figure 10.10: Four benzene (C_6H_6) rings coloured in orange, green, blue, and maroon, respectively, in (a) can be connected via fusion, as outlined by the purple bonds in (b)). This attachment of the four benzene skeletons results in a pyrene ($C_{16}H_{10}$) skeleton, coloured in brown, in (c).

- Furthermore, we introduce the definition of a linear skeleton in Axiom MSD-2 as one where the groups found in the skeleton are all chains.

$$\forall x\,(linearskeleton(x) \equiv (skeleton(x) \wedge \forall g(((group(g) \wedge mol(g,x)) \supset chain(g))))). \tag{MSD-2}$$

For example, we could consider two pentane chains (C_5H_{12}) that are connected via spiro that form a nonane (C_9H_{20}) skeleton.

(a) **(b)** **(c)**

Figure 10.11: Two pentane chains (C_5H_{12}), coloured in orange and green in (a), can be connected via spiro, outlined by the purple atom in (b). This attachment forms a nonane (C_9H_{20}) skeleton, coloured in brown, in (c).

10.3 A Mereology on Skeletons

In addition to the aforementioned properties of skeletons, we are also interested in using this notion of skeleton to indicate how skeletons can vary depending on the decomposition of a molecule. Because we allow various *decompositions* of molecules, we also must permit the notion that multiple skeletons can be formed from the combination of primitive functional groups with other groups or other atoms.

In Axiom MS-5, if two groups are *distinct*, there is a skeleton that contains both groups.

$$\forall x \forall y \, ((group(x) \wedge group(y) \wedge (x \neq y)) \supset \exists s((skeleton(s) \wedge mol(x, s) \wedge mol(y, s)))). \tag{MS-5}$$

We also need to indicate that, for *distinct* skeletons x and y, there are atoms, bonds, or groups that are contained in one skeleton, but not the other. This is outlined in Axiom MS-6.

$$\forall x \forall y \, ((skeleton(x) \wedge skeleton(y) \wedge (x \neq y)) \supset \exists z((mol(z, x) \wedge \neg mol(z, y)))). \tag{MS-6}$$

Additionally, in Axiom MSD-3, we introduce a parthood relation called $part(x, y)$ that outlines how two skeletons x and y are part of each other *if and only if* all elements found inside one skeleton are also found in the other skeleton. This is exemplified with Figure 10.5, where s_3 is composed of s_1 and the skeleton for g_3. We can state that "s_1 is part of s_3."

$$\forall x \forall y \, (part(x, y) \equiv (skeleton(x) \wedge skeleton(y) \wedge \forall z((mol(z, x) \supset mol(z, y))))). \tag{MSD-3}$$

As of writing, we currently do *not* know which mereology corresponds to molecules of the ontology – it would be more appropriate to select a mereology once we examine how chemical reactions cause changes in the structure of molecules. Future work would be to determine whether other classical extensional mereology relations from [Lew91] and [Var16] are required (such as the proper part, overlap, underlap, and sum relations), as the *mereological sums* of components of the skeleton may change depending on the decomposition (especially with esters). It is currently an open question as to which classical mereology is synonymous with the mereology on skeletons.

10.4 Summary

In this chapter, we have put everything together: we are now able to represent molecules and aggregate composite functional groups together as elements of the domain. With respect to the requirements presented in Chapter 3, we have completely satisfied Requirement (**R-5**), as we can now represent the skeletons of molecules. We have now completed the presentation of the ontology; in the next part of the dissertation, we discuss the verification of the MoSt ontology with theories found in geometry.

Part II

Verification of MoSt

Chapter 11

Grouping Axioms into Theories

Before we go into the technical details of the verification process, we can collect the axioms presented in Part I into first-order theories and their respective hierarchies. In order to do this, we present existing notions from logic about the organization of first-order theories.

11.1 First-Order Theories & Their Organization

Prior to discussing what it means for a set of theories or ontologies to be in a *hierarchy*, we adopt the following definitions from [End72] for the terms *first-order theory* and *signature*. In MoSt, recall that the axioms are written in first-order logic and are intended to be used with a first-order theorem prover, Prover9. Since we presented the axioms pedagogically in Part I, it makes sense to group the axioms into sets of first-order theories according to their pedagogical presentation.

> **Definition 11.1: First-Order Theory (Adopted from [End72])**
> *A first-order theory is a set of first-order sentences that are closed under logical entailment.*

> **Definition 11.2: Signature (Adopted from [End72])**
> *The signature, or the non-logical lexicon, of a first-order theory T is denoted by $\Sigma(T)$. It is the set of all constant symbols, function symbols, and relation symbols that are used in T. The language of T, denoted by $\mathcal{L}(T)$, is the set of first-order formulae that only use the non-logical symbols in the signature $\Sigma(T)$.*

In the context of MoSt, our presentation of the ontology in Part I outlines how we have organized the axioms into theories in this section; we have the following theories presented in Figure 11.1:

- T_{most_graph} contains axioms that pertain to molecular graph theory (atoms and bonds),

- T_{most_group} contains axioms that pertain to functional groups,
- $T_{most\_group\_definitions}$ contains the definitions for functional group concepts (such as *ring*, *chain*, and so forth),
- $T_{most\_attachment}$ contains axioms for the attachment relations (spiro, tether, fusion),
- $T_{most\_attachment\_definitions}$ contains the definitions for the attachment relations,
- $T_{most\_ringbond}$ contains the ringbond axioms,
- $T_{most\_ringbond\_definitions}$ contains the definitions for ringbonds,
- $T_{most\_betweenness}$ contains the cyclic ordering axioms,
- $T_{most\_skeleton}$ contains axioms that describe the properties of skeletons, and
- $T_{most\_skeleton\_definitions}$ contains the definitions needed for skeletons.
- T_{most} does not contain any new axioms, but is merely a theory that groups axioms of various theories within the hierarchy together. We will be using this theory later in Chapter 14 to discuss model-theoretic notions of attachment.

Recall Figure 4.1, where we presented the semantic conditions for the ontology. We have reproduced the figure here with the corresponding theory names in Figure 11.1. As well, we note that the theories listed belong to two different hierarchies: a hierarchy corresponding to chemical graph theory notions presented in Chapter 5.1 ($\mathbb{H}^{molecular\_graph}$) and a hierarchy containing all of the molecular structure notions presented in Chapters 6 to 10 ($\mathbb{H}^{most}$).

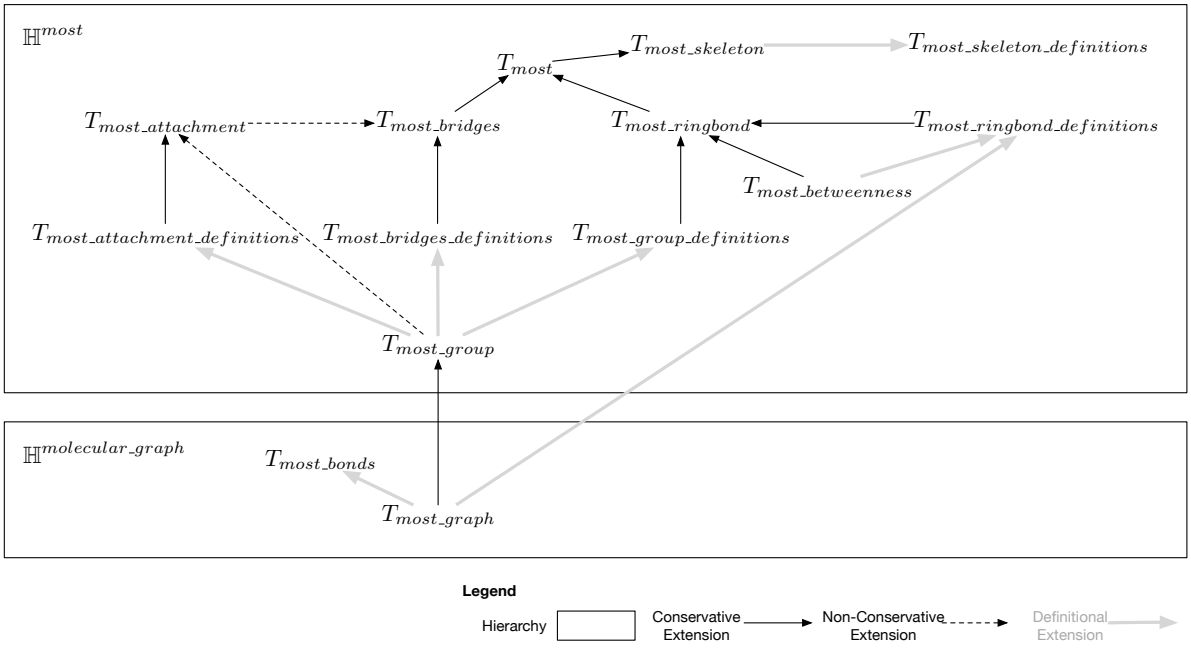


Figure 11.1: Updated MoSt hierarchy with theory names. Boxes indicate hierarchies, solid black arrows indicate conservative extension, dotted arrows indicate non-conservative extension, and solid grey arrows indicate definitional extension.

While grouping axioms into theories is useful, we are also interested in the logical relationships between the theories – we see these relationships outlined in Figure 4.1, where the theory at the head of the arrow is an *extension* of the theory at the tail of the arrow. The notions of *extension*, *conservative extension*, *non-conservative extension*, and *definitional extension* are relationships about first-order theories defined in [Grü+12] and are discussed below.

Definition 11.3: Extension (Adopted from [Grü+12])
Let T_1 and T_2 be two first-order theories such that $\Sigma(T_1) \subseteq \Sigma(T_2)$.
T_2 is an extension of T_1 iff for any sentence $\sigma \in \mathcal{L}(T_1)$,

$$\text{if } T_1 \models \sigma, \text{ then } T_2 \models \sigma.$$

Definition 11.4: Conservative Extension (Adopted from [Grü+12])
Let T_1 and T_2 be two first-order theories such that $\Sigma(T_1) \subseteq \Sigma(T_2)$.
T_2 is a conservative extension of T_1 iff for any sentence $\sigma \in \mathcal{L}(T_1)$,

$$T_2 \models \sigma \text{ iff } T_1 \models \sigma.$$

Definition 11.5: Non-Conservative Extension (Adopted from [Grü+12])
Let T_1 and T_2 be two first-order theories such that $\Sigma(T_1) \subseteq \Sigma(T_2)$.
T_2 is a non-conservative extension of T_1 iff T_2 is an extension of T_1 and there exists a sentence $\sigma \in \Sigma(T_1)$ where

$$T_1 \not\models \sigma \text{ and } T_2 \models \sigma.$$

Definition 11.6: Definitional Extension (Adopted from [GM03; Grü99])
Let T_1 and T_2 be two first-order theories such that $\Sigma(T_1) \subseteq \Sigma(T_2)$.
T_2 is a definitional extension of T_1 iff every sentence in T_2 is a definition with respect to T_1 or a sentence entailed by T_1. This means that the nonlogical lexicon in T_2 can be completely defined in terms of T_1, and adds no new expressive power to T_1.

With MoSt, in Figure 11.1, we can see examples of the following relationships between theories:

- T_{most_bonds} is a definitional extension of T_{most_graph},
- T_{most_group} conservatively extends T_{most_graph},
- $T_{most\_attachment}$ non-conservatively extends T_{most_group} and conservatively extends $T_{most\_attachment\_definitions}$,
- $T_{most\_skeleton}$ conservatively extends $T_{most\_ringbond}$,
- $T_{most\_skeleton\_definitions}$ is a definitional extension of $T_{most\_skeleton}$, and so forth.

With these relationships in mind, we revisit the idea of a first-order ontology: it is a set of first-order sentences (axioms) that characterize a first-order theory, which is the *closure* of the ontology's axioms under logical entailment. Two ontologies O_1 and O_2 that use the same non-logical lexicon Σ have logically equivalent theories if all sentences σ expressed in Σ can be represented as follows:

$$O_1 \models \sigma \Leftrightarrow O_2 \models \sigma$$

With these definitions, we are able to order sets of theories that are expressed in the same signature. We adopt the definition used in [Grü+12] to describe the notion of a hierarchy; as we can see in Figure 11.1, T_{most_graph} is located in a different hierarchy than T_{most_group}, as the latter conservatively extends the former.

Definition 11.7: Hierarchy (Adopted from [Grü+12])
A hierarchy $\mathbb{H} = \langle \mathcal{H}, \leq \rangle$ *is a partially ordered, finite set of theories* $\mathcal{H} = T_1, \ldots, T_n$*, such that:*

1. *For all* i *and* j, $\Sigma(T_i) = \Sigma(T_j)$,
2. $T_i \leq T_j$ *iff* T_j *is an extension of* T_i,
3. $T_i < T_j$ *iff* T_j *is a non-conservative extension of* T_i.

All theories in a particular hierarchy share the same set of non-logical symbols, and are ordered by non-conservative extensions, such that the extensions restrict the set of models of which the theory extends. This ordering relation allows us to say that a theory T_i is *stronger* than a theory T_j if T_i is a non-conservative extension of T_j.

Definition 11.8: Root Theory (Adopted from [Grü+12])
A theory T *in a hierarchy is a root theory iff it does not non-conservatively extend any other theory in the same hierarchy.*

In the context of MoSt, T_{most_graph} is the root theory of $\mathbb{H}^{molecular\_graph}$ and T_{most_group} is the root theory of $\mathbb{H}^{most}$, respectively. From Figure 11.1, we can see that these two theories do not non-conservatively extend any other theories in the hierarchy.

11.2 Relationships Between Hierarchies

An ontology repository like COLORE allows us to examine the network of meta-theoretic relations defined between the theories found in the repository. These relationships allow us to compare the theories easily rather than simply examining the models generated from the axioms. This comparison enables us to determine whether one theory is stronger, weaker, equivalent to, or inconsistent with another. New theories can also be defined to capture shared, or overlapping, models between two theories.

In the context of MoSt, we can see Figure 11.1 contains theories from two different hierarchies, $\mathbb{H}^{molecular\_graph}$ and $\mathbb{H}^{most}$. During the design of the ontology, we decided

to have separate hierarchies: $\mathbb{H}^{molecular\_graph}$ would contain all of the chemical graph theoretic notions that were discussed in Chapter 5, as these are established concepts in both chemistry and graph theory, where as $\mathbb{H}^{most}$ contains all of the *new* theories and notions that are introduced and discussed in this dissertation. Keeping them separate also allows us to highlight the major contributions of this work, as well as provide a more intricate discussion of how the various relations and axioms arose, inspired from the work done in [Tri92], [Coh01], and [MKH14].

We adopt the following theorem from [Grü+12] to show that a hierarchy $\mathbb{H}_1$ is not necessarily a non-conservative extension of another hierarchy $\mathbb{H}_2$, since new theories can always be added to $\mathbb{H}_2$ that are conservatively extended by theories in $\mathbb{H}_1$.

Theorem 11.2.1: Conservative Extension (Adopted from [Grü+12])
Suppose T_1 and T_2 are theories that are in different hierarchies such that $\Sigma(T_1) \subset \Sigma(T_2)$. If T_2 is a non-conservative extension of T_1, then there exists a theory T_3 such that:

- *T_2 is a conservative extension of T_3, and*
- *T_3 is compatible with the hierarchy of T_1: $\Sigma(T_3) = \Sigma(T_1)$.*

In the context of MoSt, we can see that T_{most_group} conservatively extends T_{most_graph} even though both theories are in different hierarchies. In order to compare ontologies that are axiomatized in different and disjoint non-logical lexicons, there is a need to translate a theory from one lexicon to the other while preserving the original semantics of the relations. The following definitions are adopted and adapted from [End72] and [Grü+12].

Definition 11.9: Interpretation (Adopted from [End72; Grü+12])
An interpretation π of a theory T_1 with the signature $\Sigma(T_1)$ into a theory T_2 with the signature $\Sigma(T_2)$ is a function on the set of non-logical symbols of $\Sigma(T_1)$ and formulae in $\mathcal{L}(T_1)$, such that

1. *π assigns to $\forall$ a formula $\pi_\forall$ of $\mathcal{L}(T_2)$, in which at most the variable v_1 occurs free, such that*
$$T_2 \models (\exists v_1)\pi_\forall$$
2. *π assigns to each n-place relation symbol P a formula π_P of $\mathcal{L}(T_2)$, in which at most the variable $v_1, \ldots, v_n$ occur free.*
3. *For any sentence $\sigma \in \mathcal{L}(T_1)$,*
$$T_1 \models \sigma \Rightarrow T_2 \models \pi(\sigma)$$

The mapping π is an interpretation of T_1 if it preserves the theorems of T_1; we say that "T_1 is interpretable in T_2."

Definition 11.10: Faithful Interpretation (Adopted from [End72; Grü+12])
An interpretation π of a theory T_1 into a theory T_2 is a faithful interpretation,

if and only if, for any sentence $\sigma \in \mathcal{L}(T_1)$,

$$T_1 \not\models \sigma \Rightarrow T_2 \not\models \pi(\sigma)$$

Thus, the mapping π is a faithful interpretation of T_1 if it preserves satisfiability with respect to T_1. We will also refer to this by saying that "T_1 is *faithfully interpretable* in T_2." The proof for this theorem can be found in [Grü+12].

With this in mind, the definition of definable equivalence is also adopted from [Grü+12] to generalize the notion of logical equivalence between theories that do not have the same signature.

Definition 11.11: Definable Equivalence (Adopted from [Grü+12])
Two theories, T_1 *and* T_2*, are definably equivalent iff* T_1 *is faithfully interpretable in* T_2*, and* T_2 *is faithfully interpretable in* T_1*.*

An example of definably equivalent theories can be found in temporal ontologies, such as between the mathematical theories of timepoints and linear orderings axiomatized in [Hay96]. In contrast, the theory of partial orderings is interpretable in the theory of timepoints, but these two theories are *not* definably equivalent because the theory of timepoints is not interpretable in the theory of partial orderings.

Recall Figure 3.3, where we presented a schematic from [GOS09] that outlines the relationships between intended, unintended and omitted models of an ontology. With ontology verification, we want to characterize the models of an ontology up to isomorphism and to determine whether or not the ontology's models are elementarily equivalent to the intended structures of the ontology [GO11]. To do this, we require the use of representation theorems from mathematics, defined below.

Definition 11.12: Representation Theorem (Adopted from [GO11])
A class of structures $\mathfrak{M}$ *can be represented by a class of structures* $\mathfrak{N}$ *iff there is a bijection* $\varphi : \mathfrak{M} \to \mathfrak{N}$ *such that, for any* $\mathcal{M} \in \mathfrak{M}$*,* $\mathcal{M}$ *is definable in* $\varphi(\mathcal{M})$ *and* $\varphi(\mathcal{M})$ *is definable in* $\mathcal{M}$*.*

Representation theorems are proven in two parts: first, we prove every intended structure is a model of the ontology and then prove that every model of the ontology is elementary equivalent to some intended structure. This means that classes of structures for theories within an ontology are axiomatized up to elementary equivalence: theories are satisfied by any structure in the class, and any model of the theories is elementarily equivalent to a structure in the class [GO11]. As we will see in Chapter 13, we will be using incidence geometries as the class of structures to verify MoSt.

11.2.1 Reducibility of Theories

Another approach to modularity with ontologies is based on the relationship of reducibility, in which one ontology is definably equivalent to the *union* of existing modules found in different hierarchies [Grü+10; Grü+12]. We adopt the following definition of reducibility

from [Grü+12].

Definition 11.13: Reducibility (Adopted from [Grü+12])
A theory, T, is reducible to a set of theories T_1, ..., T_n iff:

- *T faithfully interprets each theory T_i, and*
- *$T_1 \cup ... \cup T_n$ faithfully interprets T.*

In the remainder of this work, we refer to the set of theories T_1, ..., T_n as the "*reduction of T*" in COLORE. From this definition, we can see that two *definably equivalent* theories are reducible to each other. A trivial example can be found in the Combined Time hierarchy[1] in COLORE, where the theory of timepoints is reducible to the theory of linear orderings, and vice versa. A non-trivial example of reducibility can be seen in the Process Specification Language (PSL) Ontology, with the PSL-Core theory, T_{psl_core}[2]. In [Grü+10], the authors show that T_{psl_core} is reducible to T_{linear}, $T_{partition}$, and $T_{graph\_incidence}$. This example illustrates how reducibility leads to the decomposition of a theory that is treated as a module within a larger ontology [Grü+10], thus we adopt the following from [Grü+12]:

Theorem 11.2.2: Reducibility (Adopted from [Grü+12])
Let T_1, ..., T_n be a set of theories such that $\Sigma(T_i) \cap \Sigma(T_j) = \emptyset$ for all $i \leq j, j \leq n, i \neq j$. A theory T is reducible to T_1, ..., T_n iff T is definably equivalent to $T_1 \cup ... \cup T_n$.

The proof for this theorem can be found in [Grü+12]. From this theorem, the following corollary is also defined:

Corollary 11.2.3:
If T_1 is definably equivalent to T_2, then T_1 is reducible to T_2.

This theorem and corollary are particularly important with ontology verification, as a theory's reducibility tells us whether the intended models of an ontology correspond to the models of a well understood theory; in the context of MoSt, we want to determine if T_{most_graph} and T_{most_group} are reducible to the mathematical theories found in COLORE.

11.2.2 Translation Definitions

In order to map concepts between ontologies, we specify the *semantic mappings* in the form of translation definitions; this definition is adopted from [Grü+12] and [End72].

Definition 11.14: Translation Definition (Adopted from [Grü+12; End72])
Let T_0 be a theory with the signature $\Sigma(T_0)$ and T_1 be a theory with the signature $\Sigma(T_1)$, such that $\Sigma(T_0) \cap \Sigma(T_1) = \emptyset$. If there is an interpretation of T_0 in T_1, then there exists a set of sentences that axiomatizes the mapping, called a

[1] `http://colore.oor.net/combined_time/`
[2] `http://colore.oor.net/psl_core/psl_core.clif`

translation definition, in the language of $L_0 \cup L_1$ of the form:

$$(\forall \bar{x}) p_i(\bar{x}) \equiv \Phi \bar{x}$$

where $p_i(\bar{x})$ is a relation symbol in L_0 and $(\Phi \bar{x})$ is a formula in L_1 whose only free variables are $\bar{x}$.

From [Szc77], translation definitions can be considered to be an axiomatization of the interpretation of T_0 into T_1, where they conservatively extend T_0 and definitionally extend T_1.

11.2.3 Proving Relationships Between Theories

We utilized a semi-automated procedure to verify theories with the aid of the Prover9 and Mace4 software applications[3]. Prover9 is an automated theorem prover for first-order logic that uses resolution to prove goal sentences which are entailed by the inputted theory; Mace4 is a finite model generator that complements Prover9 since it searches for countermodels of the inputted goal.

To prove relationships between two theories found in different hierarchies, we adopt the methodology used in [Grü+12] to determine the definable equivalence of theories. Suppose Δ_{12} and Δ_{21} are the translations for T_1 into T_2 and T_2 into T_1, respectively. To *verify* that two theories, T_1 and T_2, are *definably equivalent*, we carry out the following reasoning problems:

1. $T_1 \cup T_2 \cup \Delta_{12}$ is consistent,
2. $T_1 \cup \Delta_{12} \models T_2$,
3. $T_1 \cup T_2 \cup \Delta_{21}$ is consistent, and
4. $T_2 \cup \Delta_{21} \models T_1$.

If all four reasoning problems produce successful results, then it means that theories T_1 and T_2 are definably equivalent. This means that T_1 and T_2 are alternative axiomatizations of the *same* set of models. If steps 1 or 3 fail, then the translation definitions between the theories are inconsistent and the two theories have two disjoint sets of models; this means that they are not translatable into one another. If step 2 fails, then it indicates that T_1 may be weaker than T_2; likewise, if step 4 fails, then T_2 may be weaker than T_1. For these 'weaker' theory scenarios, one theory is strictly weaker than the other if it is possible to find a definably equivalent theory of the stronger theory in the core hierarchy of the weaker theory, and show that it non-conservatively extends the weaker theory.

From Figure 11.1, we can see that the axioms of MoSt can be collected into the theories listed below. Two hierarchies are used to categorize the theories: the Molecular Graph Hierarchy ($\mathcal{H}^{molecular\_graph}$) contains axioms that only describe chemical graphs

[3]Prover9 and Mace4 are available via `http://www.cs.unm.edu/~mccune/mace4/`.

(previously presented in Chapter 5), whereas the MoSt Hierarchy ($\mathcal{H}^{most}$) contains axioms that expand upon chemical graphs (presented in Chapters 6 to 10). Each of the hierarchies consist of various theories that are briefly described in the subsequent sections.

11.3 Signature of MoSt

For reference, a signature (lexicon) for MoSt has also been included below. Tables 11.1 and 11.2 summarize the key terms in the lexicon of the theories of MoSt.

11.4 The $\mathbb{H}^{molecular\_graph}$ Hierarchy

In the $\mathbb{H}^{molecular\_graph}$ hierarchy, theories pertaining to chemical graph concepts discussed in Chapter 5 can be found here. As seen in Figure 11.2, there are only two theories in this hierarchy, as the axioms found in this hierarchy only deal with basic connectivity concepts between atoms and bonds, and definitions for bond types.

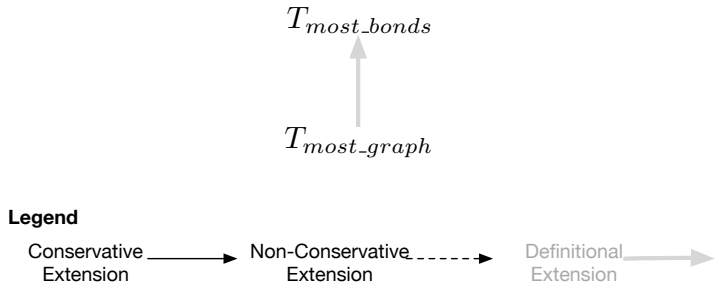


Figure 11.2: Theories found within the $\mathbb{H}^{molecular\_graph}$ hierarchy.

11.4.1 T_{most_graph}

T_{most_graph}[4] contains axioms that describes the notion of chemical graphs in terms of atoms and bonds. The axioms are summarized and grouped below:

$$\forall x\,(mol(x,x)). \tag{MG-1}$$

$$\forall x \forall y\,(mol(x,y) \supset\ mol(y,x)). \tag{MG-2}$$

$$\forall x \forall y\,((mol(x,y) \wedge atom(x) \wedge atom(y)) \supset\ (x = y)). \tag{MG-3}$$

$$\forall x \forall y\,((mol(x,y) \wedge bond(x) \wedge bond(y)) \supset\ (x = y)). \tag{MG-4}$$

[4] `http://colore.oor.net/molecular_graph/most_graph.clif`

Table 11.1: Signature of MoSt.

Extension	Predicate	Description
$T_{most\_attachment}$	$attached(g1, g2)$	group g_1 is attached to group g_2
	$bridge(x)$	x is a bridge
	$fused(g1, g2)$	groups g_1 and g_2 are fused together
	$intersects(b1, b2)$	two bonds b_1 and b_2 intersect
	$spiro(g1, g2, a)$	groups g_1 and g_2 are spiroed together
	$tether(g1, g2, b)$	groups $g1$ and $g2$ groups g_1 and g_2 are tethered together
$T_{most\_betweenness}$	$between(x, y, z)$	x is between y and z
	$inSkeleton(x, s)$	Atom x is in skeleton s.
	$skeleton(s)$	s is an skeleton.
T_{most_bonds}	$doublebond(b, x, y)$	b is a doublebond between atoms x and y
	$singlebond(b, x, y)$	b is a singlebond between atoms x and y
	$triplebond(b, x, y)$	b is a triplebond between atoms x and y
$T_{most\_bridges}$	$bridge(a, g1, g3)$	an atom a in group $g1$ is in the bridge of a moleucle if the group is fused along multiple bonds of another group $g3$
	$bridgehead(a, g1, g2)$	an atom a in rings $g1$ and $g2$ participates in a bond with three or more non-hydrogen skeletal atoms
	$fusedAtom(a, g1, g2)$	atom a_1 is in both groups g_1 and g_2 and is part of a fused bond
	$multiply\_fused(g1, g2)$	two groups $g1$ and $g2$ are multiply fused together along 2 or more bonds
T_{most_graph}	$atom(a)$	a is an atom.
	$bond(b)$	b is an bond.
	$mol(x, y)$	x and y are connected by the mol relation.

Table 11.2: Signature of MoSt (cont.).

Extension	Predicate	Description
T_{most_group}	$alternatinggroup(x)$	x is a group that consists of alternating single and double bonds
	$chain(x)$	x is a chain
	$end(x, y)$	x is an end of group y
	$fork(x, y)$	x is a fork of group y
	$group(g)$	g is an group.
	$ring(x)$	x is a ring
	$saturated(x)$	x is saturated
	$trivialgroup$	x is a trivial group
	$unsaturated(x)$	x is unsaturated
$T_{most\_ringbond}$	$ringBond(b)$	bond b is in a ring
$T_{most\_scaffold}$	$linearskeleton(x)$	x is a linear skeleton
	$part(x, y)$	x is part of y
	$scaffold(x)$	x is scaffold
$T_{most\_skeleton}$	$fusedSkeleton(s)$	A fused skeleton s only contains fused rings.
	$skeleton(s)$	s is an skeleton.

$$\forall x \forall y \forall z \forall b\, ((atom(x) \wedge atom(y) \wedge atom(z) \wedge bond(b) \wedge mol(x,b) \wedge mol(y,b) \wedge mol(z,b)) \supset ((x = y) \vee (x = z) \vee (y = z))). \tag{MG-5}$$

$$\forall x\, (atom(x) \supset \exists b \exists y((atom(y) \wedge bond(b) \wedge mol(x,b) \wedge mol(y,b)))). \tag{MG-6}$$

$$\forall b\, (bond(b) \supset \exists x \exists y((atom(x) \wedge atom(y) \wedge mol(x,b) \wedge mol(y,b)))). \tag{MG-7}$$

$$\forall b\, (bond(b) \supset \exists x \exists y(\forall z_1 \forall z_2(((atom(x) \wedge atom(y) \wedge atom(z_1) \wedge atom(z_2) \wedge mol(z_1,b) \wedge mol(z_2,b)) \supset ((z_1 = x) \vee (z_1 = y) \vee (z_2 = x) \vee (z_2 = y)))))). \tag{MG-8}$$

$$\forall b\, (bond(b) \supset \exists x \exists y((atom(x) \wedge atom(y) \wedge mol(x,b) \wedge mol(y,b) \wedge (y \neq x)))). \tag{MG-9}$$

11.4.2 T_{most_bonds}

Similarly, T_{most_bonds}[5] contains axioms that describe the various bond types. The axioms are summarized below:

$$\forall b\,(bond(b) \supset \exists x \exists y((atom(x) \wedge atom(y) \wedge (x \neq y) \wedge (singlebond(b,x,y) \vee doublebond(b,x,y) \vee triplebond(b,x,y))))). \quad \text{(MB-1)}$$

$$\forall b \forall x \forall y\,(singlebond(b,x,y) \equiv (bond(b) \wedge atom(x) \wedge atom(y) \wedge mol(x,b) \wedge mol(y,b) \wedge \forall b_2(((bond(b_2) \wedge mol(x,b_2) \wedge mol(y,b_2)) \supset (b = b_2))))). \quad \text{(MB-2)}$$

$$\forall b \forall x \forall y\,(doublebond(b,x,y) \equiv \exists b_2((atom(x) \wedge atom(y) \wedge bond(b_2) \wedge mol(x,b) \wedge mol(y,b) \wedge mol(x,b_2) \wedge mol(y,b_2) \wedge (b \neq b_2)))). \quad \text{(MB-3)}$$

$$\forall b \forall x \forall y\,(triplebond(b,x,y) \equiv \exists b_2 \exists b_3((atom(x) \wedge atom(y) \wedge bond(b_2) \wedge bond(b_3) \wedge mol(x,b) \wedge mol(y,b) \wedge mol(x,b_2) \wedge mol(y,b_2) \wedge mol(x,b_3) \wedge mol(y,b_3) \wedge (b \neq b_2) \wedge (b_2 \neq b_3)))). \quad \text{(MB-4)}$$

11.5 The $\mathbb{H}^{most}$ Hierarchy

In the $\mathbb{H}^{most}$ hierarchy, we expand upon the signature of T_{most_graph} by introducing new concepts (rings, chains, functional groups, ends, forks, and so forth), new relations for attachment (spiro, tether, fusion), and additional supplementary relations for connectivity (betweenness, ringbonds, and so forth). The relationships between theories in this hierarchy are depicted in Figure 11.3.

11.5.1 T_{most_group}

T_{most_group}[6] contains the axioms and definitions for the basic concepts in MoSt, including the notion of functional groups as elements of the domain. The axioms are summarized below:

$$\forall x\,(atom(x) \supset \neg(bond(x) \vee group(x))). \quad \text{(M-1)}$$

$$\forall x\,(bond(x) \supset \neg group(x)). \quad \text{(M-2)}$$

[5] http://colore.oor.net/molecular_graph/definitions/most_bonds.clif
[6] http://colore.oor.net/most/most_group.clif

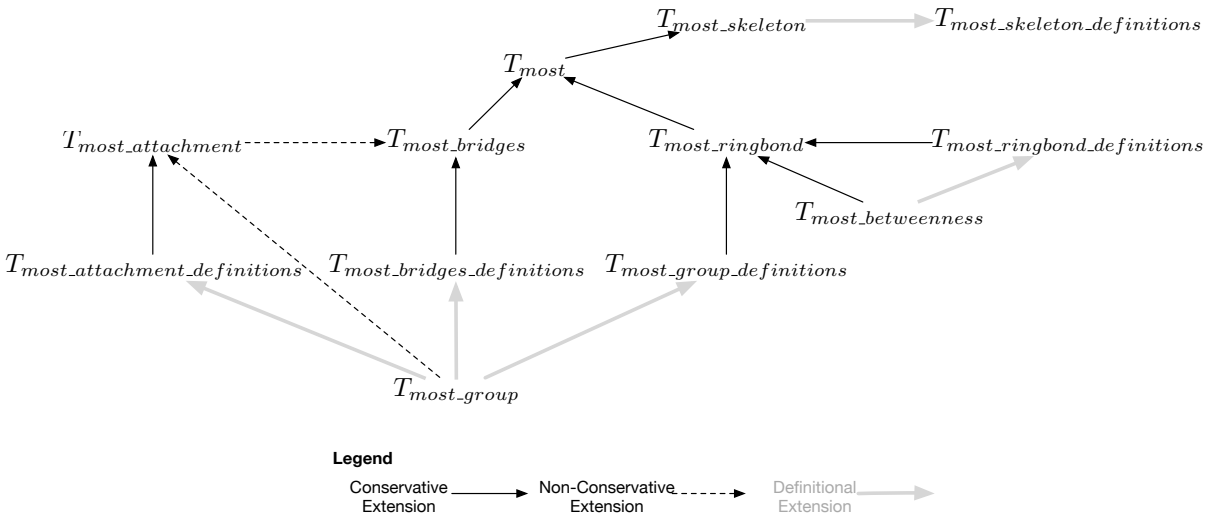


Figure 11.3: Theories found within the $\mathbb{H}^{most}$ hierarchy.

$$\forall x \forall y \, ((mol(x,y) \wedge group(x) \wedge group(y)) \supset (x = y)). \tag{M-3}$$

$$\forall x \, (group(x) \supset \exists a((atom(a) \wedge mol(a,x)))). \tag{M-4}$$

$$\forall x \, (atom(x) \supset \exists y((group(y) \wedge mol(x,y)))). \tag{M-5}$$

$$\begin{aligned} \forall x \, (atom(x) \supset \exists b \exists y \exists z((bond(b) \wedge group(y) \wedge atom(z) \wedge \\ (x \neq z) \wedge mol(x,b) \wedge mol(z,b) \wedge mol(z,y)))). \end{aligned} \tag{M-6}$$

$$\begin{aligned} \forall b \forall g \forall x \forall y \, ((atom(x) \wedge atom(y) \wedge (x \neq y) \wedge \\ bond(b) \wedge group(g) \wedge mol(x,b) \wedge \\ mol(y,b) \wedge mol(b,g)) \supset mol(x,g)). \end{aligned} \tag{M-7}$$

$$\begin{aligned} \forall x \forall y \forall z \, ((atom(x) \wedge bond(y) \wedge group(z) \wedge \\ mol(x,y) \wedge mol(y,z)) \supset mol(x,z)). \end{aligned} \tag{M-8}$$

$$\forall y \, (group(y) \supset \exists x((atom(x) \wedge mol(x,y)))). \tag{M-9}$$

$$\begin{aligned} \forall x \forall y \forall b \forall g \, ((atom(x) \wedge atom(y) \wedge bond(b) \wedge group(g) \wedge \\ (x \neq y) \wedge mol(x,g) \wedge mol(y,g) \wedge mol(x,b) \wedge \\ mol(y,b)) \supset mol(b,g)). \end{aligned} \tag{M-10}$$

$$\forall b \forall g\ ((bond(b) \wedge group(g) \wedge \neg mol(b,g)) \supset \exists a((atom(a) \wedge mol(a,b) \wedge \neg mol(a,g)))). \tag{M-11}$$

$$\forall x \forall y \forall z \forall w\ ((group(x) \wedge end(y,x) \wedge end(z,x) \wedge end(w,x)) \supset ((y = z) \vee (y = w) \vee (z = w))). \tag{M-12}$$

$$\begin{aligned}\forall a_1 \forall a_2 \forall a_3 \forall a_4 \forall g \forall b_1 \forall b_2 \forall b_3\ (&group(g) \wedge atom(a_1) \wedge atom(a_2) \wedge atom(a_3) \wedge \\ &atom(a_4) \wedge mol(a_1,g) \wedge mol(a_2,g) \wedge \\ &mol(a_3,g) \wedge mol(a_4,g) \wedge bond(b_1) \wedge \\ &mol(b_1,g) \wedge mol(a_1,b_1) \wedge mol(a_2,b_1) \wedge \\ &bond(b_2) \wedge mol(b_2,g) \wedge mol(a_1,b_2) \wedge \\ &mol(a_3,b_2) \wedge bond(b_3) \wedge mol(b_3,g) \wedge \\ &mol(a_1,b_3) \wedge mol(a_4,b_3) \supset\ (a_4 = a_3) \vee \\ &(a_2 = a_3) \vee (a_4 = a_2)).\end{aligned} \tag{M-13}$$

11.5.2 $T_{most\_group\_definitions}$

Furthermore, definitions are included in $T_{most\_group\_definitions}$[7]

$$\begin{aligned}\forall x \forall y\ (end(x,y) \equiv (&atom(x) \wedge group(y) \wedge mol(x,y) \wedge \\ &\forall b_1 \forall b_2 \forall w \forall z(((bond(b_1) \wedge bond(b_2) \wedge \\ &atom(w) \wedge atom(z) \wedge mol(x,b_1) \wedge \\ &mol(z,b_1) \wedge mol(x,b_2) \wedge mol(w,b_2) \wedge \\ &mol(z,y) \wedge mol(w,y)) \supset (w = z))))).\end{aligned} \tag{MD-1}$$

$$\begin{aligned}\forall x \forall y\ (fork(x) \equiv (&atom(x) \wedge \exists b_1 \exists b_2 \exists b_3 \exists a_1 \exists a_2 \exists a_3((atom(a_1) \wedge \\ &atom(a_2) \wedge atom(a_3) \wedge bond(b_1) \wedge bond(b_2) \wedge \\ &bond(b_3) \wedge (a_1 \neq a_2) \wedge (a_2 \neq a_3) \wedge (a_1 \neq a_3) \wedge \\ &(b_1 \neq b_2) \wedge (b_2 \neq b_3) \wedge (b_1 \neq b_3) \wedge \\ &mol(x,b_1) \wedge mol(a_1,b_1) \wedge mol(x,b_2) \wedge \\ &mol(a_2,b_2) \wedge mol(x,b_3) \wedge mol(a_3,b_3))))).\end{aligned} \tag{MD-2}$$

$$\forall x\ (chain(x) \equiv (group(x) \wedge \exists y((end(y,x) \wedge \forall w(((atom(w) \wedge mol(w,x)) \supset \neg fork(w))))))). \tag{MD-3}$$

$$\forall x\ (ring(x) \equiv (group(x) \wedge \forall y(((atom(y) \wedge mol(y,x)) \supset (\neg end(y,x) \wedge \neg fork(y)))))). \tag{MD-4}$$

[7] http://colore.oor.net/most/definitions/most_group_definitions.clif

$$\forall x\,(saturated(x) \equiv (group(x) \wedge \forall b \forall y \forall z(((bond(b) \wedge atom(y) \wedge mol(y,b) \wedge mol(z,b) \wedge mol(y,x) \wedge mol(z,x)) \supset singlebond(b,y,z))))). \quad \text{(MD-5)}$$

$$\forall x\,(unsaturated(x) \equiv (group(x) \wedge \exists b \exists y \exists z(((atom(y) \wedge bond(b) \wedge mol(y,b) \wedge mol(z,b) \wedge mol(y,x) \wedge mol(z,x)) \supset doublebond(b,y,z))))). \quad \text{(MD-6)}$$

$$\forall x \forall y\,(alternatinggroup(x) \equiv ((group(x) \wedge atom(y) \wedge mol(x,y)) \supset \exists a_1 \exists a_2 \exists b_1 \exists b_2((atom(a_1) \wedge atom(a_2) \wedge bond(b_1) \wedge bond(b_2) \wedge mol(y,b_1) \wedge mol(a_1,b_1) \wedge doublebond(b_2,y,a_2) \wedge mol(a_1,x) \wedge mol(a_2,x))))). \quad \text{(MD-7)}$$

$$\forall x\,(trivialgroup(x) \equiv (group(x) \wedge \forall y \forall z(((atom(y) \wedge atom(z) \wedge mol(y,x) \wedge mol(z,x)) \supset (y = z))))). \quad \text{(MD-8)}$$

11.5.3 $T_{most\_attachment}$

$T_{most\_attachment}$[8] contains the axioms and definitions for the attachment relations in MoSt (spiro, tether, fusion). The axioms are summarized below:

$$\forall a_1 \forall a_2 \forall a_3 \forall b \forall g_1 \forall g_2\,((atom(a_1) \wedge atom(a_2) \wedge atom(a_3) \wedge group(g_1) \wedge group(g_2) \wedge (g_1 \neq g_2) \wedge bond(b) \wedge mol(b,g_1) \wedge mol(b,g_2) \wedge spiro(g_1,g_2,a_1) \wedge spiro(g_1,g_2,a_2) \wedge spiro(g_1,g_2,a_3)) \supset ((a_1 = a_2) \vee (a_1 = a_3) \vee (a_2 = a_3))). \quad \text{(MA-1)}$$

$$\forall a_1 \forall a_2 \forall a_3 \forall g_1 \forall g_2\,((mol(a_1,g_1) \wedge mol(a_1,g_2) \wedge mol(a_2,g_1) \wedge mol(a_2,g_2) \wedge (a_1 \neq a_2) \wedge (g_1 \neq g_2) \wedge between(a_1,a_3,a_2)) \supset (mol(a_3,g_1) \wedge mol(a_3,g_2))). \quad \text{(MA-2)}$$

$$\forall x \forall y\,((group(x) \wedge group(y) \wedge (x \neq y)) \supset \exists a((atom(a) \wedge mol(a,x) \wedge \neg mol(a,y)))). \quad \text{(MA-3)}$$

[8] http://colore.oor.net/most/most_attachment.clif

$$\forall g_1 \forall g_2 \forall a_1 \forall a_2 \; ((group(g_1) \wedge group(g_2) \wedge (g_1 \neq g_2) \wedge atom(a_1) \wedge atom(a_2) \wedge (a_1 \neq a_2) \wedge mol(a_1, g_1) \wedge mol(a_2, g_2)) \supset \neg mol(a_1, g_2) \wedge \neg mol(a_2, g_1))). \quad \text{(MA-4)}$$

11.5.4 $T_{most\_attachment\_definitions}$

Furthermore, definitions for the attachment relations are included in $T_{most\_attachment\_definitions}$[9]

$$\forall g_1 \forall g_2 \; (fused(g_1, g_2) \equiv (group(g_1) \wedge group(g_2) \wedge (g_1 \neq g_2) \wedge \exists a_1 \exists a_2 \exists b((atom(a_1) \wedge atom(a_2) \wedge (a_1 \neq a_2) \wedge mol(a_1, b) \wedge mol(a_2, b) \wedge mol(a_1, g_1) \wedge mol(a_1, g_2) \wedge mol(a_2, g_1) \wedge mol(a_2, g_2) \wedge mol(b, g_1) \wedge mol(b, g_2))))). \quad \text{(MAD-1)}$$

$$\forall g_1 \forall g_2 \forall b \; (tether(g_1, g_2, b) \equiv (group(g_1) \wedge group(g_2) \wedge bond(b) \wedge (g_1 \neq g_2) \wedge \exists a_1 \exists a_2 ((atom(a_1) \wedge atom(a_2) \wedge mol(a_1, g_1) \wedge mol(a_2, g_2) \wedge mol(a_1, b) \wedge mol(a_2, b) \wedge \neg mol(b, g_1) \wedge \neg mol(b, g_2))))). \quad \text{(MAD-2)}$$

$$\forall g_1 \forall g_2 \forall a \; (spiro(g_1, g_2, a) \equiv (group(g_1) \wedge group(g_2) \wedge atom(a) \wedge (g_1 \neq g_2) \wedge \exists b_1 \exists b_2((bond(b_1) \wedge bond(b_2) \wedge mol(a, g_1) \wedge mol(a, g_2) \wedge mol(a, b_1) \wedge mol(a, b_2) \wedge mol(b_1, g_1) \wedge \neg mol(b_1, g_2) \wedge mol(b_2, g_2) \wedge \neg mol(b_2, g_1))))). \quad \text{(MAD-3)}$$

$$\forall g_1 \forall g_2 \; (attached(g_1, g_2) \equiv (fused(g_1, g_2) \vee \exists a(spiro(g_1, g_2, a)) \vee \exists b(tether(g_1, g_2, b)))). \quad \text{(MAD-4)}$$

[9] http://colore.oor.net/most/definitions/most_attachment_definitions.clif

11.5.5 $T_{most\_bridges}$

$T_{most\_bridges}$[10] contains the axioms and definitions for the basic concepts in MoSt. The axioms are summarized below:

$$\exists a_1 \forall g_1 \forall g_2 \forall a_2 \, ((spiro(g_1, g_2, a_1) \wedge spiro(g_1, g_2, a_2)) \supset (a_1 = a_2)). \quad \text{(MBR-1)}$$

$$\exists b_1 \forall g_1 \forall g_2 \forall b_2 \, ((tether(g_1, g_2, b_1) \wedge tether(g_1, g_2, b_2)) \supset (b_1 = b_2)). \quad \text{(MBR-2)}$$

$$\forall g_1 \forall g_2 \, (fused(g_1, g_2) \supset \exists a_1 \exists a_2((fusedAtom(a_1, g_1, g_2) \wedge fusedAtom(a_2, g_1, g_2) \wedge (a_1 \neq a_2)))). \quad \text{(MBR-3)}$$

$$\forall a_1 \forall a_2 \forall a_3 \forall g_1 \forall g_2 \, ((bridgehead(a_1, g_1, g_2) \wedge bridgehead(a_2, g_1, g_2) \wedge bridgehead(a_3, g_1, g_2)) \supset ((a_1 = a_2) \vee (a_1 = a_3) \vee (a_2 = a_3))). \quad \text{(MBR-4)}$$

11.5.6 $T_{most\_bridges\_definitions}$

Furthermore, definitions for bridge-related concepts are included in $T_{most\_bridges\_definitions}$[11]

$$\forall g_1 \forall g_2 \, (multiply\_fused(g_1, g_2) \equiv (group(g_1) \wedge group(g_2) \wedge (g_1 \neq g_2) \wedge \exists b_1 \exists b_2(bond(b_1) \wedge bond(b_2) \wedge (b_1 \neq b_2) \wedge mol(b_1, g_1) \wedge mol(b_1, g_2) \wedge mol(b_2, g_1) \wedge mol(b_2, g_2)))). \quad \text{(MBRD-1)}$$

$$\forall a_1 \forall g_1 \forall g_2 \, (fusedAtom(a_1, g_1, g_2) \equiv (atom(a_1) \wedge group(g_1) \wedge group(g_2) \wedge mol(a_1, g_1) \wedge mol(a_1, g_2) \wedge (g_1 \neq g_2) \wedge \exists b_1 \exists b_2((bond(b_1) \wedge bond(b_2) \wedge mol(a_1, b_1) \wedge mol(a_1, b_2) \wedge mol(b_1, g_1) \wedge mol(b_2, g_2))))). \quad \text{(MBRD-2)}$$

$$\forall a \forall g_1 \forall g_2 \, (bridge(a, g_1, g_2) \equiv (atom(a) \wedge group(g_1) \wedge group(g_2) \wedge mol(a, g_1) \wedge mol(a, g_3) \wedge \forall g_3(((group(g_3) \wedge mol(a, g_3)) \supset (g_1 = g_3))) \wedge multiply\_fused(g_1, g_2))). \quad \text{(MBRD-3)}$$

[10] http://colore.oor.net/most/most_bridges.clif

[11] http://colore.oor.net/most/definitions/most_bridges_definitions.clif

$$\begin{aligned}\forall a \forall g_1 \forall g_2\ (bridgehead(a, g_1, g_2) \equiv\ & (atom(a) \wedge group(g_1) \wedge group(g_2) \wedge \\ & (g_1 \neq g_2) \wedge \exists b \exists b_1 \exists b_2 ((bond(b) \wedge bond(b_1) \wedge \\ & bond(b_2) \wedge (b_1 \neq b_2) \wedge (b \neq b_1) \wedge (b \neq b_2) \wedge \\ & mol(a, b) \wedge mol(a, b_1) \wedge mol(a, b_2) \wedge \\ & mol(b, g_1) \wedge mol(b, g_2) \wedge mol(b_1, g_1) \wedge \\ & \neg mol(b_1, g_2) \wedge \neg mol(b_2, g_1) \wedge \\ & mol(b_2, g_2))))).\end{aligned} \tag{MBRD-4}$$

11.5.7 $T_{most\_ringbond}$

$T_{most\_ringbond}$[12] contains a singular axiom for the notion of a ringbond in MoSt.

$$\forall b\ (ringBond(b) \supset \exists g((mol(b, g) \wedge ring(g)))). \tag{MRB-1}$$

11.5.8 $T_{most\_ringbond\_definitions}$

Furthermore, the definition for ringbond is found in $T_{most\_ringbond\_definitions}$[13]

$$\begin{aligned}\forall b\ ringBond(b) \equiv\ & (bond(b) \wedge \exists a_1 \exists a_2 ((atom(a_1) \wedge atom(a_2) \wedge \\ & mol(a_1, b) \wedge mol(a_2, b) \wedge \neg(\exists a_3 ((atom(a_3) \wedge \\ & (betweenMol(a_3, a_1, a_2) \vee \\ & betweenMol(a_1, a_2, a_3)))))))).\end{aligned} \tag{MRBD-1}$$

11.5.9 $T_{most\_betweenness}$

$T_{most\_betweenness}$[14] contains the axioms and definitions for the cyclic ordering concepts in MoSt. The axioms are summarized below:

$$\forall x \forall y \forall z\ (betweenMol(x, y, z) \supset (atom(x) \wedge atom(y) \wedge atom(z))). \tag{MBTWN-1}$$

$$\begin{aligned}\forall x \forall y \forall z \forall s\ ((& skeleton(s) \wedge atom(x) \wedge atom(y) \wedge atom(z) \wedge \\ & mol(x, s) \wedge mol(y, s) \wedge mol(z, s) \wedge \\ & betweenMol(x, y, z)) \supset betweenMol(z, y, x)).\end{aligned} \tag{MBTWN-2}$$

[12] http://colore.oor.net/most/most_ringbond.clif
[13] http://colore.oor.net/most/definitions/most_ringbond_definitions.clif
[14] http://colore.oor.net/most/most_betweenness.clif

$$\forall x \forall y \forall z \forall s\ ((skeleton(s) \wedge atom(x) \wedge atom(y) \wedge atom(z) \wedge mol(x,s) \wedge mol(y,s) \wedge mol(z,s) \wedge betweenMol(y,x,z) \wedge betweenMol(x,y,z)) \supset (x = y)). \quad \text{(MBTWN-3)}$$

$$\forall x \forall y \forall z \forall w \forall s\ ((skeleton(s) \wedge atom(x) \wedge atom(y) \wedge atom(z) \wedge mol(x,s) \wedge mol(y,s) \wedge mol(z,s) \wedge mol(w,s) \wedge betweenMol(y,x,z)) \supset (betweenMol(y,x,w) \vee betweenMol(z,x,w))). \quad \text{(MBTWN-4)}$$

$$\forall a_1 \forall a_2 \forall a_3 \forall g_1 \forall g_2\ ((mol(a_1,g_1) \wedge mol(a_1,g_2) \wedge mol(a_2,g_1) \wedge mol(a_2,g_2) \wedge (a_1 \neq a_2) \wedge (g_1 \neq g_2) \wedge betweenMol(a_1,a_3,a_2)) \supset (mol(a_3,g_1) \wedge mol(a_3,g_2))). \quad \text{(MBTWN-5)}$$

11.5.10 $T_{most\_skeleton}$

Finally, $T_{most\_skeleton}$[15] contains the axioms and definitions for the skeleton concepts in MoSt. The axioms are summarized below:

$$\forall x\ (skeleton(x) \supset \neg((atom(x) \vee bond(x) \vee group(x)))). \quad \text{(MS-1)}$$

$$\forall x \forall y\ ((mol(x,y) \wedge skeleton(x) \wedge skeleton(y)) \supset (x = y)). \quad \text{(MS-2)}$$

$$\forall s\ (skeleton(s) \supset \exists g((group(g) \wedge mol(g,s)))). \quad \text{(MS-3)}$$

$$\forall g_1 \forall s\ ((skeleton(s) \wedge group(g_1) \wedge mol(g_1,s)) \supset \exists a_1 \exists a_2 \exists b \exists g_2((atom(a_1) \wedge atom(a_2) \wedge bond(b) \wedge group(g_2) \wedge (g_1 \neq g_2) \wedge mol(a_1,g_1) \wedge mol(a_2,g_2) \wedge mol(a_1,b) \wedge mol(a_2,b)))). \quad \text{(MS-4)}$$

$$\forall x \forall y\ ((group(x) \wedge group(y) \wedge (x \neq y)) \supset \exists s((skeleton(s) \wedge mol(x,s) \wedge mol(y,s)))). \quad \text{(MS-5)}$$

$$\forall x \forall y\ ((skeleton(x) \wedge skeleton(y) \wedge (x \neq y)) \supset \exists z((mol(z,x) \wedge \neg mol(z,y)))). \quad \text{(MS-6)}$$

[15] http://colore.oor.net/most/most_skeleton.clif

11.5.11 $T_{most\_skeleton\_definitions}$

Furthermore, definitions for skeleton-related concepts are included in $T_{most\_skeleton\_definitions}$[16]

$$\forall s\,(fusedSkeleton(s) \equiv (skeleton(s) \wedge \forall g_1 \forall g_2(((group(g_1) \wedge mol(g_1,s) \wedge group(g_2) \wedge mol(g_2,s) \wedge (g_1 \neq g_2)) \supset (ring(g_1) \wedge ring(g_2) \wedge fused(g_1,g_2)))))). \quad \text{(MSD-1)}$$

$$\forall x\,(linearskeleton(x) \equiv (skeleton(x) \wedge \forall g(((group(g) \wedge mol(g,x)) \supset chain(g))))). \quad \text{(MSD-2)}$$

$$\forall x \forall y\,(part(x,y) \equiv (skeleton(x) \wedge skeleton(y) \wedge \forall z((mol(z,x) \supset mol(z,y))))). \quad \text{(MSD-3)}$$

11.5.12 T_{most}

In Chapter 14, we will be discussing techniques for model construction using the models of MoSt. We mention here that the theory called T_{most}[17] in Figure 11.1 refers to the union of $T_{most\_bridges}$ and $T_{most\_ringbond}$, and imports their axioms.

$$T_{most} \equiv T_{most\_bridges} \cup T_{most\_ringbond}$$

11.6 Summary

In this chapter, we have grouped all of the MoSt axioms into theories; moving forward, we will discuss the verification process, and notions of incidence structures and the graph-theoretic concepts used in the next chapter.

We note here that we only present the verification results for T_{most_graph} and T_{most_group}, as these are the two core theories that serve as the foundations for the rest of the ontology. For $T_{most\_skeleton}$, the verification of this theory is intended for future work as it is currently unclear which type of mereology corresponds to skeletons.

[16] http://colore.oor.net/most/definitions/most_skeleton_definitions.clif
[17] http://colore.oor.net/most/most.clif

Chapter 12

Methodology

Since chemical graphs are the underlying structures for the work in this thesis, we discuss *ontology verification* and the methodology adopted to ensure the ontology is accurate and correct in relation to incidence structures found in mathematics.

12.1 Ontology Verification

Ontology verification is concerned with the relationship between the intended models of an ontology and the models of the axiomatization of the ontology. In particular, we want to characterize the models of an ontology up to isomorphism and determine whether or not these models are equivalent to the intended models of the ontology. In practice, the verification of an ontology is achieved by demonstrating that it is *synonymous*[1] with a theory whose models have already been characterized up to isomorphism.

We characterize the semantics of an ontology as a set of *intended structures*[2]. We specify these structures with well-understood mathematical theories to determine whether the axiomatization of an ontology matches its intended models; these theories include partial orderings, lattices, incidence structures, geometries, and algebra [Grü11; Grü+10]. If an ontology's axiomatization contains *unintended* models, then it is possible to find sentences that are entailed by the intended models, but these sentences are not provable from the axioms of the ontology. Such models provide barriers to semantic interoperability between software systems and may prevent the entailment of sentences [Grü11; Grü+10].

By verifying an ontology, we would like to characterize its models up to isomorphism to determine whether or not these models are *equivalent* to the intended structures of the ontology [Grü11]. To do this, we utilize the mathematical notion of *representation theorems*, where we prove that every intended structure is a model of the ontology and that every model of the ontology is *elementary equivalent* to some intended structure[3].

[1] Two ontologies T and O with disjoint signatures are synonymous iff there exists a theory S with signature $\Sigma(S) = \Sigma(T) \cup \Sigma(O)$ such that S is a definitional extension of both T and O.

[2] Adopted from [Grü11] for the ontology, an intended structure is a set of structures that characterizes the semantics of an ontology's terminology.

[3] For additional reading on ontology verification, we refer the reader to [Grü+10].

In this work, we leverage work done by mathematicians for bipartite/tripartite/quadripartite incidence structures, orderings, lattices, algebra and graph theory to verify the axioms of MoSt.

Recall Figure 3.3 in Chapter 3.3 where we presented four additional requirements to ensure the equivalence of models and molecules with MoSt; through verification, we explore the mappings between MoSt models with incidence structures found in mathematics, and discuss how the *intended* models of MoSt do correspond to existing and chemically-feasible molecules.

In Figure 12.1, we illustrate what is called the *Synonymy Approach* in [Aam16], which assumes that an ontology is correct *if and only if* there is a one-to-one correspondence between the models of the ontology and the intended models.

Definition 12.1: Verifiably Correct (Adopted from [Aam16])
An ontology T is verifiably correct with respect to $\mathfrak{M}^{intended}$ iff there exists a bijective mapping $\varphi : Mod(T) \to \mathfrak{M}^{intended}$ such that, for all $\mathcal{M} \in Mod(T)$, $\varphi(\mathcal{M})$ and $\mathcal{M}$ have a common definitional expansion.

As we will see in Chapter 13, we present two representation theorems for T_{most_graph} and T_{most_group} that contain bijective mappings for their synonymous incidence geometry theories.

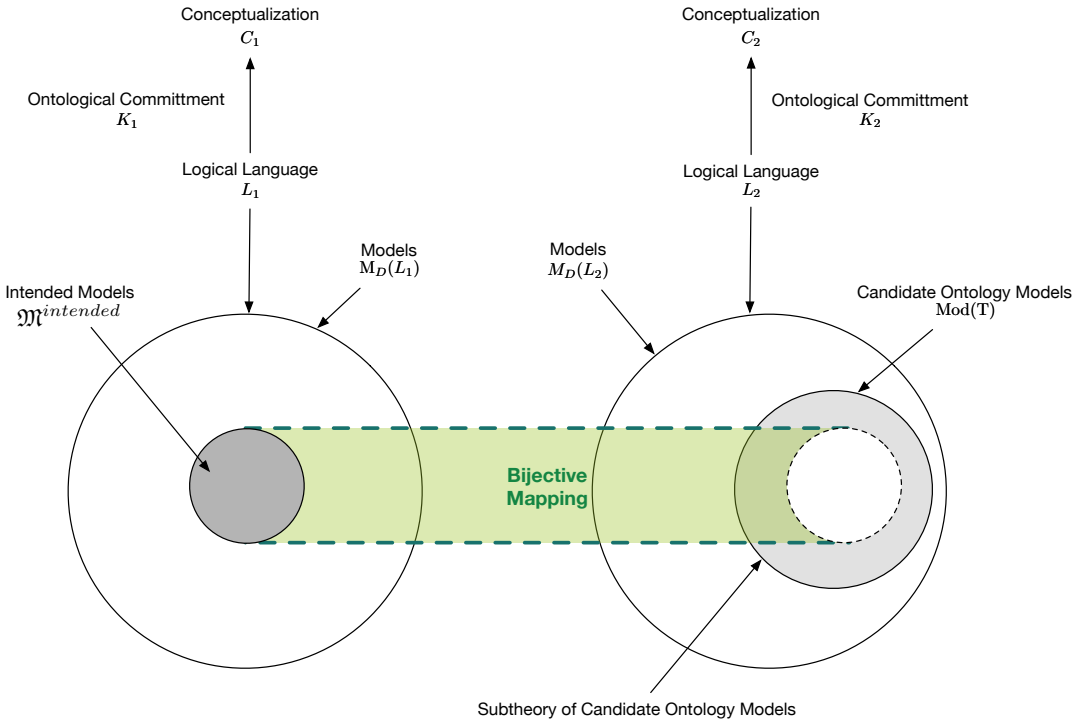


Figure 12.1: Verifiably correct mappings between the ontology's models ($\mathcal{M}$) and the intended models $\mathfrak{M}^{intended}$. The ontology T is verifiably correct with respect to $\mathfrak{M}^{intended}$ if T is synonymous with $Th(\mathfrak{M}^{intended})$. The bijective mapping is outlined in green. Adopted from [Aam16] and [GOS09].

12.2 Incidence Structures

Given that the design of MoSt draws its basis from graph theory, something we noticed during the design process was that the axiomatization falls in line with graph-theoretic structures that resemble incidence structures found in mathematics. It was apparent that some intuitions about incidence structures carried over in the design of the ontology; examples include atoms being incident with bonds, which resembles the notion of points being incident with lines.

Incidence structures are presented and heavily discussed in [Moo17] and [Bue95]; in this work, they form the basis of mathematical models with which we compare the models of MoSt. Incidence geometry arises from the points, lines, and planes of elementary geometry [Bue95]. They have provided the basis for the verification work done with time points and intervals [GO11], OWL-Time [Grü11], the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [CG17], and the Suggested Upper Merged Ontology (SUMO) upper ontologies [MG16]. Further, when used with verification, incidence geometries have also provided new techniques for ontology modularization to encourage reuse of first-order ontologies [GHK11; Grü+10].

An incidence structure, or incidence system, consists of objects with incidence relations between these objects. Usually the objects are points, lines, planes, etc. The most basic incidence structure is the point-line incidence structure $(\mathfrak{B}, \mathfrak{L}, I)$ (often abbreviated $(\mathfrak{B}, \mathfrak{L})$), where $\mathfrak{B}$ and $\mathfrak{L}$ are sets of points and lines, respectively, with an incidence relation $I \subseteq \mathfrak{B} \times \mathfrak{L}$ that indicates which point-line pairs are incident [Moo17]. An example of point-line (bipartite) incidence is shown in Figure 12.2, where points p_1 and p_2 are incident with line l_1, and p_2 is also incident with line l_2.

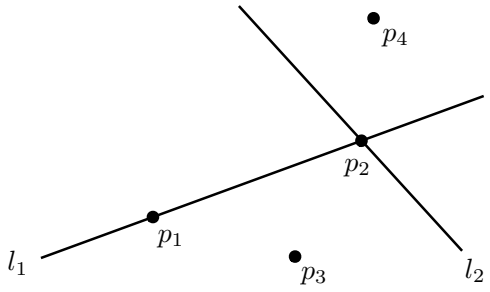


Figure 12.2: Example of point-line incidence, where points p_1 and p_2 are incident with line l_1, and p_2 is also incident with line l_2.

Similarly, Figure 12.3 outlines an example of point-line-plane (tripartite) incidence: a line l_1 intersects the plane q at point p_3, a point p_4 is incident with l_1, while a line l_2 is incident with the plane q and contains two points p_1 and p_2. We can generalize this notion for *k-partite* incidence structures by using Definition 12.2 from [CG14].

Definition 12.2: k-partite Incidence (Adopted from [CG14])
A k-partite incidence structure is a tuple $\mathbb{I} = \langle \Omega_1, ..., \Omega_k, \mathbf{in} \rangle$*, where* $\Omega_1, ..., \Omega_k$ *are sets such that* $\Omega_i \cap \Omega_j = \emptyset, i \neq j$ *and* $\mathbf{in} \subseteq (\bigcup_{i \neq j} \Omega_i \times \Omega_j)$*. Two elements of* $\mathbb{I}$ *that are related by* **in** *are called incident. The neighbourhood of an element is the set of elements which are incident with it:* $N(\mathbf{x}) = \{\mathbf{y} : \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbf{in}\}$.

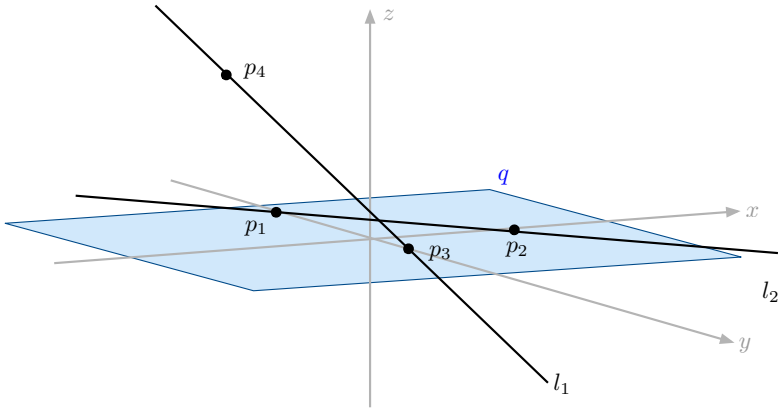


Figure 12.3: Example of point-line-plane incidence, where a line l_1 intersects the plane q at point p_3, a point p_4 is incident with l_1, while a line l_2 is incident with the plane q and contains two points p_1 and p_2.

Since we are leveraging existing theories in COLORE for the verification of MoSt, the axioms for bipartite incidence structures can be found in the $\mathbb{H}^{bipartite\_incidence}$ Hierarchy[4], and similarly for the tripartite incidence axioms in the $\mathbb{H}^{triipartite\_incidence}$ Hierarchy[5]. Following the origins of these structures from geometry, we refer to the elements of the sets of *bipartite incidence* structures as points and lines, and the sets of *tripartite incidence* structures as points, lines, and planes.

12.3 Graph-Theoretic Notions

In addition to using bipartite and tripartite incidence structures, we also revisit graph-theoretic notions from mathematics. Since MoSt was designed with graph theory in mind, we also realized that graph-theoretic concepts also aid us in the verification of the ontology. Before we can outline the basis for the verification of MoSt, we briefly summarize the concepts and definitions found in graph theory: graph, path, cycle, bridge, and block. These concepts are defined in a foundational graph theory book, [Die12], published by Springer in their series *Graduate Texts in Mathematics*, vol. 173.

We revisit the definition of a graph from Chapter 5 (recall Definition 5.1, which we have reprinted below for reference).

> **Definition 12.3: Graph (Adopted from [Die12])**
> *A graph is a pair $G = (V, E)$ of sets such that $E \subseteq [V]^2$; thus, the elements of E are 2-element subsets of V. The elements of V are the vertices (or nodes, or points) of the graph G, the elements of E are its edges (or lines). See Figure 5.1.*

As graphs form the basis of MoSt, we also present the formal definition of a subgraph in Definition 12.4. Subgraphs are also important as they also have influenced the axiom-

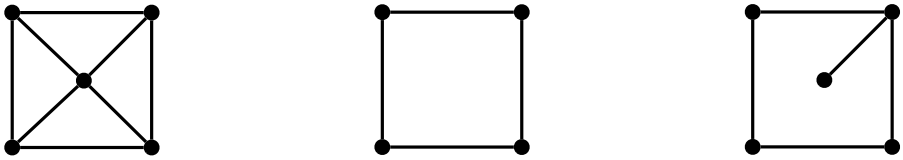
[4] http://colore.oor.net/bipartite_incidence/
[5] http://colore.oor.net/tripartite_incidence/

atization of the ontology: we want to be able to isolate subgraphs, such as the rings and paths, of the underlying chemical graph of a molecule. We have seen subgraphs in Part I where we discuss how we axiomatize primitive functional groups in a molecule: these groups (rings, chains) are subgraphs of the underlying chemical graph. We illustrate the differences between a graph G, and its subgraphs G' and G'' in Figure 12.4.

Definition 12.4: Subgraph (Adopted from [Die12])
Let $G = (V, E)$ and $G' = (V', E')$ be two graphs. We set $G \cup G' := (V \cup V', E \cup E')$ and $G \cap G' := (V \cap V', E \cap E')$.

If $V' \subseteq V$ and $E' \subseteq E$, then G' is a subgraph of G: $G' \subseteq G$ (and conversely, G is a supergraph of G').

If $G' \subseteq G$ contains all of the edges $xy \in E$ with $x, y \in V'$, then G' is an induced subgraph of G. This can also be referred to as V' induces or spans G' in G: $G' =: G[V']$.



(a) G **(b)** G' **(c)** G''

Figure 12.4: A graph G in (a) with its subgraphs G' in (b) and G'' in (c). G' in (b) is an induced subgraph of G, whereas G'' in (c) is not an induced subgraph of G.

We also utilize the notions of paths and cycles in graphs, as they correspond to chains and rings in MoSt, respectively. In Figure 12.5, we have a graph G with a path P in G.

Definition 12.5: Path (Adopted from [Die12])
A path is a non-empty graph $P = (V, E)$ of the form:

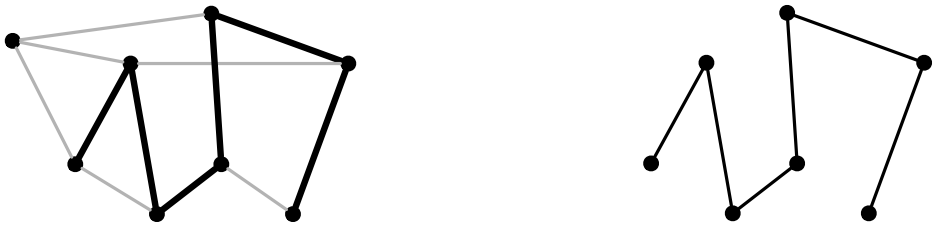
$$V = \{x_0, x_1, ..., x_k\}$$

$$E = \{x_0x_1, x_1x_2, ..., x_{k-1}x_k\}$$

where the x_i are all distinct. The vertices x_0 and x_k are linked by P and are called its ends; the vertices $x_1, ..., x_{k-1}$ are the inner vertices of P. The number of edges of a path is its length, and the path of length k is denoted by P^k. (Alternatively, we can consider the path P_k as a path on k vertices.) A path length is allowed to be zero, thus $P^0 = P_1 = K_1$ (a complete graph on 1).

Definition 12.6: Induced Path (Adopted from [Die12])
Following Definition 12.5, an induced path of an undirected graph G is a path that is an induced subgraph of G.

Similarly, a cycle in a graph is denoted by a cyclic sequence of vertices, as shown in



(a) G **(b)** $P = P^6$

Figure 12.5: A path with length 6, $P = P^6$, in G. The path P is highlighted in thicker lines in (a) and is shown on its own in (b). We can also state that P is an induced path of G.

Figure 12.6, where a cycle is shown with its induced cycles and chord.

Definition 12.7: Cycle (Adopted from [Die12])
If $P = x_0, ..., x_{k-1}$ is a path and $k \geq 3$, then the graph $C := P + x_k - 1x_0$ is called a cycle. As with paths, we denote a cycle by its cyclic sequence of vertices: a cycle C may be written as $x_0x_1...x_{k-1}x_0$. The length of a cycle is its number of edges (or vertices); the cycle of length k is called a k-cycle and is denoted by C^k.

An edge which joins two vertices of a cycle but is not itself an edge of the cycle is called a chord of the cycle.

Definition 12.8: Induced Cycle (Adopted from [Die12])
Following Definition 12.7, an induced cycle in a graph G is a cycle in G that forms a subgraph that has no chords.

(a) C^8 in G **(b)** $P = P^6$

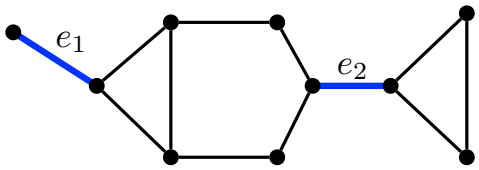
Figure 12.6: A cycle C^8 in graph G is shown in thicker lines in (a). In (b), the chord xy is highlighted in dotted blue lines, along with the induced cycles C^4 and C^6 in magenta and orange, respectively.

With these definitions, we are able to find the induced cycles and induced paths that correspond to rings and chains in the underlying graph of a molecule using the spanning tree axioms that were presented in Chapter 9.2. This is especially important as fundamental cycles introduced in Definition 9.3 form the cycle basis for a graph: they are also induced cycles, but are further restricted by the requirement that an edge not in the spanning tree of the graph is used to form a cycle in the graph. With respect to

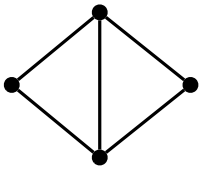
MoSt, we utilize the spanning tree of the underlying chemical graph to find the simple cycles of a graph using the notion of ringbonds (which were presented in Chapter 9.5).

We also make note that, in graph theory, the notion of a bridge is **different** than the notion of bridges found in chemistry. In graph theory, bridges are also called isthmuses, cut-edges, or cut-arcs. We make this distinction in Definition 12.9, where a bridge is an edge that does not lie on any cycle in a graph, and illustrate this in Figure 12.7. We contrast this with bridges in chemistry where bridges are indeed part of the graph and correspond to multiply-fused bonds in the graph.

Definition 12.9: Bridge (Graph Theory) (Adopted from [Die12])
Let $G = (V, E)$ be a graph. Bridges in a G are edges that do not lie on any cycle. A graph is said to be bridgeless if it contains no bridges.



(a) A graph G with bridges e_1 and e_2, highlighted in blue.



(b) A bridgeless graph G.

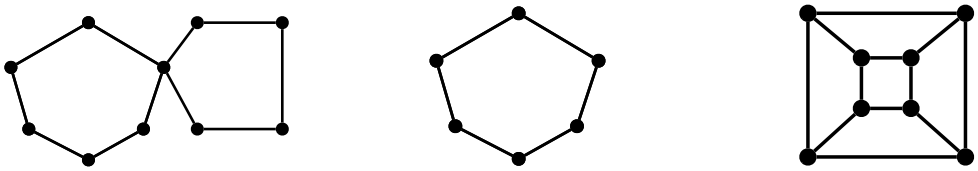
Figure 12.7: Examples of bridges in graphs.

We also introduce the notion of *k-connectedness* in graphs, as 2-connected graphs become especially important for our decomposition of a graph into its blocks, as well as a requirement for the verification process. A graph is *connected* if, for any two vertices $x, y \in V(G)$, there is a path whose endpoints are x and y. Further, a connected graph G is called *2-connected* if, for every vertex $x \in V(G)$, $G-x$ is connected (see Definition 12.11). In simpler terms, a graph is k-connected if it cannot be disconnected by removing less than k vertices.

Definition 12.10: k-connected (Adopted from [Die12])
A graph G is called k-connected if $|G| > k$ and $G-X$ is connected for every set $X \subseteq V$ with $|X| < k$. A 1-connected graph is called connected, a 2-connected graph is called biconnected, and a 3-connected graph is called triconnected.

Definition 12.11: 2-connected (Adopted from [Die12])
A graph G is called 2-connected if, for every vertex, $x \in V(G)$, $G - x$ is connected.

With these graph-theoretic notions in mind, recall the notion of a spanning tree (Definition 9.2). The spanning tree is essential in our verification process as the tree helps us identify the blocks found in a graph, which then are used to identify simple cycles and chains in the decomposition of the graph (to map them to rings and chains, respectively). We introduce the notion of a block below in Definition 12.12.



(a) 1-connected **(b)** 2-connected **(c)** 3-connected

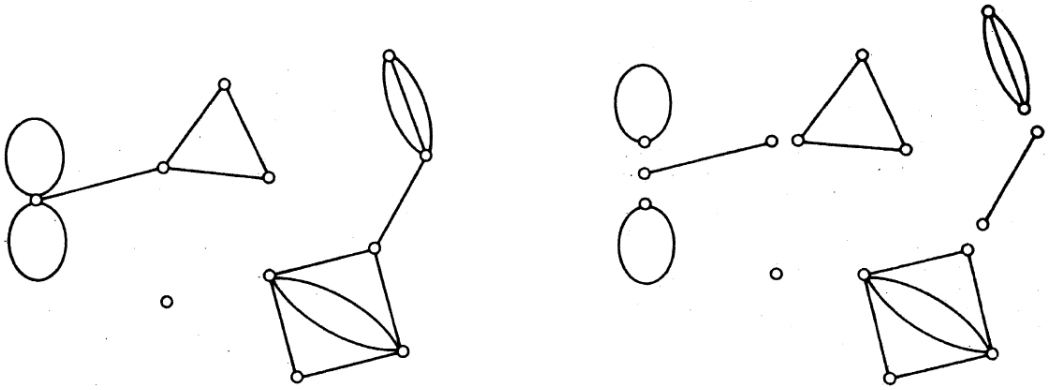
Figure 12.8: Examples of k-connected graphs: (a) is a 1-connected graph, (b) is a 2-connected graph, and (c) is a 3-connected graph.

Definition 12.12: Block (Adopted from [Die12])
A maximal connected subgraph without a cutvertex is called a block. Every block of a graph G is either a maximal 2-connected subgraph, or a bridge (with its ends), or an isolated vertex.

With the notion of blocks in mind, we can state the following theorem, as all of the graphs we deal with in the ontology are all *connected* graphs:

Theorem 12.3.1: Block Decomposition (from [BM76])
Every graph has a block decomposition: every graph is the union of its blocks.

We illustrate a graph G and its blocks in Figure 12.9.



(a) A graph G. **(b)** The blocks of G.

Figure 12.9: A graph G and its blocks. (Figure 3.3 in [BM76])

Now that we have covered the graph-theoretic notions needed for the verification (and model construction discussed in Chapter 14), we note that there is a correspondence between the models of incidence geometries and the block decomposition of a graph. We utilize Theorem 12.3.1 later in Chapter 14 along with the graph theoretic concepts when discussion the decomposition and recomposition of models. We revisit these notions when discussing techniques for composing models of MoSt.

12.4 Comments on Ontology Design & Verification

From our discussion here of verification concepts, incidence structures, and graph-theoretic notions, we can see how graph theory has influenced both the design of the ontology and its verification. As the designer of MoSt, it is much easier to picture the intended models of the ontology: we know that molecules can correspond to graphs and we can leverage the graph-theoretic notions presented in the verification (and later in the construction of first-order models). In contrast, while analyzing existing ontologies authored by other ontology designers, it is much more difficult to visualize and discern the intended models. This is particularly evident in upper ontologies that contain abstract notions of temporal constructs, parthood, constitution, and so forth, such as those found in the Basic Formal Ontology (BFO)[6], DOLCE[7], and SUMO[8]. As part of the ontology design process, designers should also specify what models of an ontology should look like, or provide examples.

12.5 Summary

In this chapter, we have discussed the incidence structures and graph-theoretic notions required for the verification of MoSt. Next, we will discuss why these notions are needed in the incidence theories used in the verification and discuss the outcomes from the technical verification results.

[6] http://basic-formal-ontology.org/
[7] http://www.loa.istc.cnr.it/old/DOLCE.html
[8] http://www.adampease.org/OP/

Chapter 13

Verifying MoSt

In this chapter, we present the technical results of the verification process, along with a discussion on the insights provided by the verification process.

From the previous chapter, we have presented a representation theorem for the models of MoSt that corresponds to tripartite incidence structures. As well, since the models of the tripartite incidence structures correspond to the block decomposition of a graph, we can utilize the Prover9 theorem prover to prove synonymy between the incidence structures and the underlying graph of a molecule.

13.1 Software Setup

To run the verification experiments using Prover9, the input ontology files were prepared from the CLIF format into Library for Automated Deduction Research (LADR) input format using the MACLEOD[1] suite of tools developed by Dr. Torsten Hahmann[2] and his students at the University of Maine. This suite of tools handles key reasoning tasks for the verification of first-order ontologies, either specified in the CLIF (`*.clif`), LADR (`*.in`), or Thousands of Problems for Theorem Provers (TPTP) (`*.tptp`) syntaxes. The tools are written in the Python programming language that interacts with the command-line version of Prover9-Mace4. In particular, the translation scripts from CLIF to LADR were used to translate the MoSt CLIF files into the Prover9 input syntax. Figure 13.1 outlines an example of the translation from these two syntaxes. Specifically, all verification experiments were run and tested on a Apple Macbook Pro running OS X Yosemite (Version 10.10.5) with a 2.5Ghz Intel Core i7 processor[3], and Python 2.7[4] and version 05B of Prover9-Mace4[5].

In order to verify a MoSt theory with the theorem prover, we need show that the axioms of T_{most_graph} entails the axioms of a graphical incidence theory in COLORE, and vice versa. This means that we need to have a mapping between the theories – we

[1] `https://github.com/thahmann/macleod`
[2] `http://tarski.ume.maine.edu/bio.shtml`
[3] `https://support.apple.com/kb/SP704`
[4] `https://www.python.org/download/releases/2.7/`
[5] `https://www.cs.unm.edu/~mccune/mace4/download/`

$$\forall x \forall y\,((group(x) \wedge group(y) \wedge (x \neq y)) \supset \exists a((atom(a) \wedge mol(a,x) \wedge \neg mol(a,y)))). \quad \text{(MA-3)}$$

(a) Axiom in first-order logic.

```
(forall (x y)
    (if (and (group x)(group y)(not (= x y)))
       (exists (a)
          (and (atom a)(mol a x)(not (mol a y))))))
```

(b) CLIF syntax.

```
all x all y (((group(x) & group(y) & (x != y)) ->
     exists a ((atom(a) & mol(a, x) & -mol(a, y))))).
```

(c) Prover9 syntax.

Figure 13.1: Examples of the CLIF and Prover9 syntaxes for first-order logic with Axiom MA-3.

do this with translation definitions, which we discussed in the previous chapter. The theorem proving exercise that needs to be done is to run Prover9 to prove every axiom of the target theory is entailed by the first theory. In the case of MoSt, we need to show that proofs can be generated in *both* directions using translation definitions Δ and Π; in other words:

- $T_{most\_theory} \cup \Delta \models$ Axiom #1 of $T_{target\_incidence\_theory}$
- $T_{most\_theory} \cup \Delta \models$ Axiom #2 of $T_{target\_incidence\_theory}$
- $T_{most\_theory} \cup \Delta \models$ Axiom #3 of $T_{target\_incidence\_theory}$
- ...
- $T_{target\_incidence\_theory} \cup \Pi \models$ Axiom #1 of T_{most_theory}
- $T_{target\_incidence\_theory} \cup \Pi \models$ Axiom #2 of T_{most_theory}
- $T_{target\_incidence\_theory} \cup \Pi \models$ Axiom #3 of T_{most_theory}
- ...

We specify the target incidence theories and the translation definitions in the sections below. All of the individual input and output files for the verification of T_{most_graph}[6] and T_{most_group}[7] can be found online on COLORE.

[6] http://colore.oor.net/molecular_graph/interprets
[7] http://colore.oor.net/most/interprets

13.2 Verification & Reduction of T_{most_graph}

With respect to T_{most_graph}, we noticed the similarities between regular graph structures found in geometry and the notions of molecular graphs (as discussed in Chapter 5). To verify T_{most_graph}, we needed a theory with the following requirements:

(**VER-1**) *Axiomatize a graph with multiple edges*: from our discussions in Chapter 5, our axiomatization of MoSt allows underlying chemical graphs to have multiple edges to ensure double and triple bonds are properly represented, which means we need axioms that axiomatize multigraphs (as seen in Definition 5.3).

(**VER-2**) *Prevent loops in graphs*: in graph theory, a loop in a graph is an edge that connects a vertex to itself. Figure 13.2 shows a graph with loops and sets of multiple edges. Since we do not have any molecules that exhibit this kind of behaviour, we do not want to allow them in the underlying molecular graph.

(**VER-3**) *Prevent isolated vertices in graphs*: in graph theory, vertices that are not part of a graph are considered as *isolated*, and the graph is then considered *disconnected*, as illustrated in Figure 13.3. With respect to MoSt, we want to prevent this from occurring because this would signify an atom is free-floating in the graph; in short, we want graphs that are *connected* with no isolated vertices.

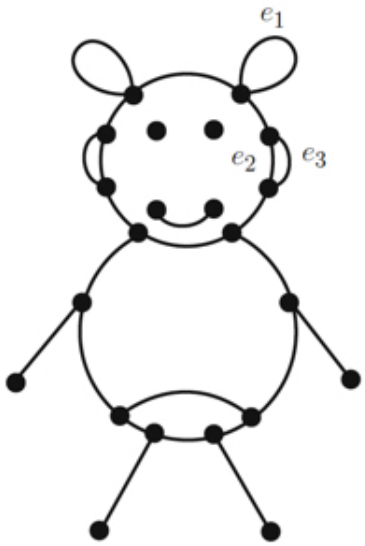


Figure 13.2: A graph where e_1 is an example of a loop and $\{e_2, e_3\}$ is a set of multiple edges. With respect to MoSt, we want to prevent loops like e_1 in the underlying graph but allow multiple edges like $\{e_2, e_3\}$. (Figure 1.3 in [BR12]).

In the $\mathbb{H}^{bipartite\_incidence}$ hierarchy of COLORE, the $T_{nonisolated\_loopless}$ theory contains axioms that cover the above requirements and is a theory that only deals with a bipartite incidence structure (points and lines). A full discussion about this theory can be found

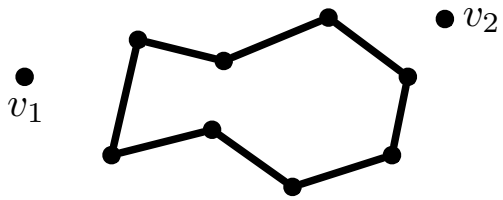


Figure 13.3: Example of an isolated graph, where v_1 and v_2 are isolated vertices.

in Chapter 5.18 ("Loopless Graphical Incidence Structures") of [Grü18]. The axioms of $T_{nonisolated\_loopless}$ can be found in COLORE[8], and are also reproduced in Figure 13.4.

$$(\forall l\ (line(l) \supset\ (\exists x(point(x) \wedge in(x,l))))). \tag{NIL-1}$$

$$(\forall x \forall y \forall z \forall l\ (point(x) \wedge point(y) \wedge point(z) \wedge line(l) \wedge in(x,l) \wedge in(y,l) \wedge in(z,l) \supset\ z = x \vee z = y \vee x = y)). \tag{NIL-2}$$

$$(\forall l\ (line(l) \supset\ (\exists x \exists y(point(x) \wedge point(y) \wedge x \neq y \wedge in(x,l) \wedge in(y,l))))). \tag{NIL-3}$$

$$(\forall x \forall y\ (in(x,y) \supset\ in(y,x))). \tag{NIL-4}$$

$$(\forall x\ (point(x) \vee line(x) \supset\ in(x,x))). \tag{NIL-5}$$

$$(\forall p\ (point(p) \supset\ \neg line(p))). \tag{NIL-6}$$

$$(\forall x \forall y\ (in(x,y) \wedge point(x) \wedge point(y) \supset\ x = y)). \tag{NIL-7}$$

$$(\forall x \forall y\ (in(x,y) \wedge line(x) \wedge line(y) \supset\ x = y)). \tag{NIL-8}$$

$$(\forall p\ (point(p) \supset\ (\exists l(line(l) \wedge in(p,l))))). \tag{NIL-9}$$

Figure 13.4: Axioms of $T_{nonisolated\_loopless}$.

Theorem 13.2.1: Reduction of T_{most_graph}
T_{most_graph} is definably equivalent to $T_{nonisolated\_loopless}$.

Proof. Let Δ_1 be the set of translation definitions:

$$(\forall x(point(x) \equiv\ atom(x))).$$
$$(\forall x(line(x) \equiv\ bond(x))).$$
$$(\forall x \forall y\ (in(x,y) \equiv\ mol(x,y))).$$

$$T_{most\_graph} \cup \Delta_1 \models T_{nonisolated\_loopless}$$

Let Π_1 be the set of translation definitions:

[8] http://colore.oor.net/bipartite_incidence/nonisolated_loopless.clif

$$(\forall x(atom(x) \equiv point(x))).$$
$$(\forall x(bond(x) \equiv line(x))).$$
$$(\forall x \forall y\,(mol(x,y) \equiv in(x,y)))$$

$$T_{nonisolated\_loopless} \cup \Pi_1 \models T_{most\_graph}$$

Using Prover9, we have shown that:

$$T_{most\_graph} \cup \Delta_1 \models T_{nonisolated\_loopless}$$

$$T_{nonisolated\_loopless} \cup \Pi_1 \models T_{most\_graph}$$

∟ □

Proofs for this theorem can be found in COLORE[9] and in Appendix D.2. Since we have shown that T_{most_graph} is synonymous with $T_{nonisolated\_loopless}$, T_{most_graph} is also consistent.

13.3 Representation Theorem for $Mod(T_{most\_graph})$

From the verification results, we can state that models of T_{most_graph} are synonymous with classes of nonisolated, loopless incidence structures found in mathematics.

From [Grü18], we can utilize the following definition for nonisolated loopless incidence structures in the context of points P, lines L, and incidence $\mathbf{I}$:

Definition 13.1: Nonisolated Loopless Incidence Structure
A nonisolated loopless incidence structure is a tuple $\mathbb{I} = \langle P, L, \mathbf{I}\rangle$ *such that:*

1. $\mathbb{I} \in \mathfrak{M}^{weak\_bipartite}$;
2. *for each* $\mathbf{l} \in L$,
$$\mid (N(\mathbf{l}) \cap P) \mid = 2$$
3. *for each* $\mathbf{p} \in P$,
$$N(\mathbf{p}) \cap L \neq \emptyset$$

$\mathfrak{M}^{nonisolated\_loopless}$ *is the class of nonisolated loopless incidence structures.*

Furthermore, $\mathbb{B} = \langle P, \mathbf{B}\rangle$ is a betweenness structure.

Using this definition, we can state the following representation theorem:

Theorem 13.3.1: Representation Theorem for T_{most_graph}
There exists a bijection

$$\varphi : Mod(T_{most\_graph}) \rightarrow \mathfrak{M}^{most\_graph}$$

[9] http://colore.oor.net/molecular_graph/interprets/

such that $\varphi(\langle \boldsymbol{M}, \boldsymbol{atom}, \boldsymbol{bond}, \boldsymbol{mol} \rangle) = \mathbb{I} \oplus \mathbb{B}$ *iff*

1. $M = P \cup L$;
2. $\langle \mathbf{a} \rangle \in \mathbf{atom}$ *iff* $\mathbf{a} \in P$;
3. $\langle \mathbf{b} \rangle \in \mathbf{bond}$ *iff* $\mathbf{b} \in L$;
4. $\langle \mathbf{x}, \mathbf{y} \rangle \in \mathbf{mol}$ *iff* $\mathbf{y} \in N(\mathbf{x})$.

13.4 Verification & Reduction of T_{most_group}

Similarly, to verify T_{most_group}, we need a theory with the following requirements, in addition to the requirements for T_{most_graph}:

(**VER-4**) *Axiomatize groups as planes*: because T_{most_group} introduces the notions of primitive functional groups in the ontology, we also need to be able to map groups to an incidence structure. In this case, we explicitly identify that these primitive functional groups are synonymous to planes in the incidence structures, which is why need a *tripartite* incidence theory.

(**VER-5**) *Axiomatize induced subgraphs (paths and cycles)*: further to the above requirement, we also want to enforce this notion that rings in a molecule are considered as induced cycles in a graph, and likewise with chains and induced paths. This is because we intend to take the underlying graph of a molecule and decompose it into its core components: rings and chains, and these rings and chains need to correspond to subgraph structures in the incidence side of things. From Definitions 12.6 and 12.8, it is clear that we need a tripartite incidence theory that handles these notions of induced paths and induced cycles.

(**VER-6**) *Axiomatize that every vertex is adjacent to another vertex*: in order to ensure we have connected, nonisolated graphs that contain points (atoms), lines (bonds), and groups (planes), we need to enforce this connection for every vertex in the graph.

As we can see here, we must consider a class of graphs that is even more restricted than $T_{nonisolated\_loopless}$ that was used to verify T_{most_graph}. This is due to the fact that we have additional attachment constraints on the structures. We outline the relationships between the incidence theories used to verify the theories of MoSt in Figure 13.5.

From Theorem 13.2.1, we already know that T_{most_graph} is definably equivalent with $T_{nonisolated\_loopless}$. Since we now need a tripartite incidence theory that contains the notion of planes, we consider the theory $T_{cycle\_path\_subgraph}$ inside the tripartite incidence hierarchy, $\mathbb{H}^{tripartite\_incidence}$; this theory axiomatizes simple cycles and simple paths as subgraphs of a graph, and also contains axioms to handle loopless graphs (from $T_{nonisolated\_loopless}$). Finally, since we need to ensure the graphs are nonisolated, we also

utilize the axioms of $T_{nonisolated\_loopless}$ to further restrict this class of graphs. The resulting theory that we need to use for the verification of T_{most_group} is $T_{cycle\_path\_subgraph\_nonisolated}$.

In the $\mathbb{H}^{tripartite\_incidence}$ hierarchy of COLORE, the $T_{cycle\_path\_subgraph\_nonisolated}$ theory contains axioms that cover the requirements above and is a theory that only deals with tripartite incidence structures (points, lines, and planes) for this restricted class of graphs. The axioms of $T_{cycle\_path\_subgraph\_nonisolated}$ can be found in COLORE[10], and are also reproduced in Figure 13.6. Axioms which are definitions of relations in $T_{cycle\_path\_subgraph\_nonisolated}$ are removed since they are not required for the verification process; these are: Axioms CPSN-19 to CPSN-34 which are listed in Appendix C.1.

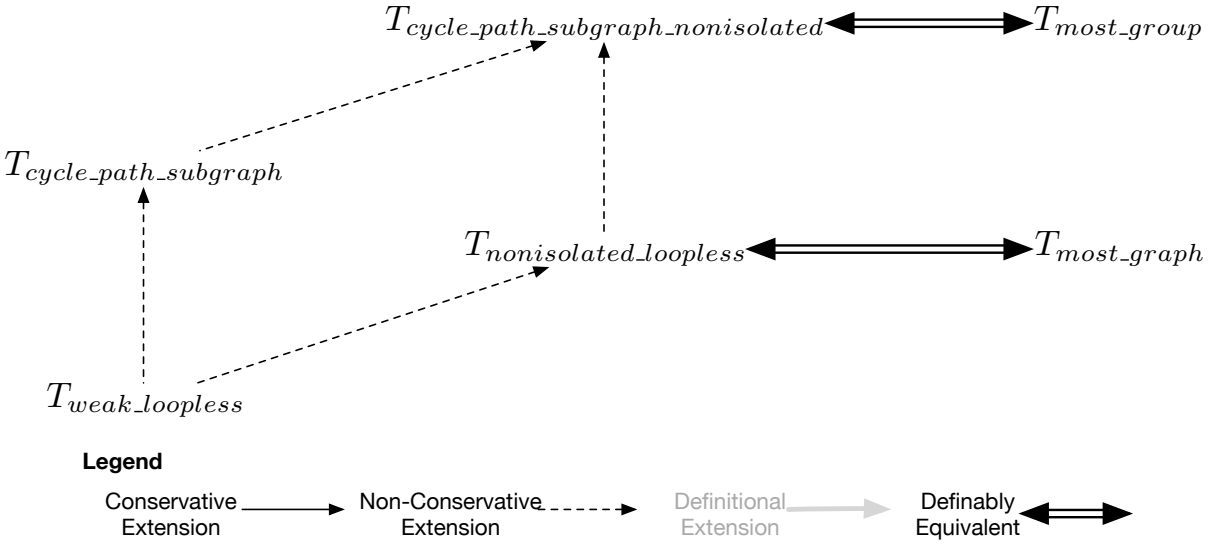


Figure 13.5: Relationships between the incidence theories that are mapped to the theories of MoSt.

[10] http://colore.oor.net/tripartite_incidence/cycle_path_subgraph_nonisolated.clif

$$(\forall x\forall y\forall z\forall w\forall q\forall l_1\forall l_2\forall l_3\ (plane(q) \wedge point(x) \wedge point(y) \wedge point(z) \wedge point(w) \wedge in(x,q) \wedge in(y,q) \wedge in(z,q) \wedge in(w,q) \wedge line(l_1) \wedge in(l_1,q) \wedge in(x,l_1) \wedge in(y,l_1) \wedge line(l_2) \wedge in(l_2,q) \wedge in(x,l_2) \wedge in(z,l_2) \wedge line(l_3) \wedge in(l_3,q) \wedge in(x,l_3) \wedge in(w,l_3) \supset\ w = z \vee y = z \vee w = y)). \quad \text{(CPSN-1)}$$

$$(\forall q\forall x\forall y\forall z\ (planar\_pendant(x,q) \wedge planar\_pendant(y,q) \wedge planar\_pendant(z,q) \supset\ x = y \vee y = z \vee z = x)). \quad \text{(CPSN-2)}$$

$$(\forall p\ (point(p) \supset\ (\exists l(line(l) \wedge in(p,l))))). \quad \text{(CPSN-3)}$$

$$(\forall l\forall q\ (line(l) \wedge plane(q) \wedge \neg in(l,q) \supset\ (\exists p(point(p) \wedge in(p,l) \wedge \neg in(p,q))))). \quad \text{(CPSN-4)}$$

$$(\forall l\ (line(l) \supset\ (\exists x\exists y(point(x) \wedge point(y) \wedge x \neq y \wedge in(x,l) \wedge in(y,l))))). \quad \text{(CPSN-5)}$$

$$(\forall x\forall y\ (in(x,y) \supset\ in(y,x))). \quad \text{(CPSN-6)}$$

$$(\forall x\ (point(x) \vee line(x) \vee plane(x) \supset\ in(x,x))). \quad \text{(CPSN-7)}$$

$$(\forall p\ (point(p) \supset\ \neg line(p))). \quad \text{(CPSN-8)}$$

$$(\forall p\ (point(p) \supset\ \neg plane(p))). \quad \text{(CPSN-9)}$$

$$(\forall p\ (plane(p) \supset\ \neg line(p))). \quad \text{(CPSN-10)}$$

$$(\forall x\forall y\ (in(x,y) \wedge point(x) \wedge point(y) \supset\ x = y)). \quad \text{(CPSN-11)}$$

$$(\forall x\forall y\ (in(x,y) \wedge line(x) \wedge line(y) \supset\ x = y)). \quad \text{(CPSN-12)}$$

$$(\forall x\forall y\ (in(x,y) \wedge plane(x) \wedge plane(y) \supset\ x = y)). \quad \text{(CPSN-13)}$$

$$(\forall p\ (point(p) \supset\ (\exists q(plane(q) \wedge in(p,q))))). \quad \text{(CPSN-14)}$$

$$(\forall l\ (line(l) \supset\ (\exists x(point(x) \wedge in(x,l))))). \quad \text{(CPSN-15)}$$

$$(\forall x\forall y\forall z\forall l\ (point(x) \wedge point(y) \wedge point(z) \wedge line(l) \wedge in(x,l) \wedge in(y,l) \wedge in(z,l) \supset\ z = x \vee z = y \vee x = y)). \quad \text{(CPSN-16)}$$

$$(\forall x\forall y\forall z\ (plane(x) \wedge line(y) \wedge point(z) \wedge in(z,y) \wedge in(y,x) \supset\ in(z,x))). \quad \text{(CPSN-17)}$$

$$(\forall q\ (plane(q) \supset\ (\exists p(point(p) \wedge in(p,q))))). \quad \text{(CPSN-18)}$$

Figure 13.6: Axioms of $T_{cycle\_path\_subgraph\_nonisolated}$.

Theorem 13.4.1: Reduction of T_{most_group}
T_{most_group} is definably equivalent to $T_{cycle\_path\_subgraph\_nonisolated}$.

Proof. Let Δ_2 be the set of translation definitions:

$$(\forall x(point(x) \equiv\ atom(x))).$$
$$(\forall x(line(x) \equiv\ bond(x))).$$
$$(\forall x(plane(x) \equiv\ group(x))).$$
$$(\forall x \forall y\ (in(x,y) \equiv\ mol(x,y))).$$

$$T_{most\_group} \cup \Delta_2 \models T_{cycle\_path\_subgraph\_nonisolated}$$

Let Π_2 be the set of translation definitions:

$$(\forall x(atom(x) \equiv\ point(x))).$$
$$(\forall x(bond(x) \equiv\ line(x))).$$
$$(\forall x(group(x) \equiv\ plane(x))).$$

$$\begin{aligned}(\forall x \forall y(end(x,y) \equiv\ & point(x) \wedge plane(y) \wedge in(x,y) \wedge (\forall l_1 \forall l_2 \forall w \forall z(line(l_1) \wedge \\ & line(l_2) \wedge point(w) \wedge point(z) \wedge in(x,l_1) \wedge in(z,l_1) \wedge in(x,l_2) \wedge \\ & in(w,l_2) \wedge in(z,y) \wedge in(w,y) \supset\ w = z)))).\end{aligned}$$

$$\begin{aligned}(\forall x \forall y(fork(x) \equiv\ & point(x) \wedge (\exists l_1 \exists l_2 \exists l_3 \exists p_1 \exists p_2 \exists p_3(point(p_1) \wedge point(p_2) \wedge \\ & point(p_3) \wedge line(l_1) \wedge line(l_2) \wedge line(l_3) \wedge p_1 \neq p_2 \wedge p_2 \neq p_3 \wedge \\ & p_1 \neq p_3 \wedge l_1 \neq l_2 \wedge l_2 \neq l_3 \wedge l_1 \neq l_3 \wedge in(x,l_1) \wedge in(p_1,l_1) \wedge \\ & in(x,l_2) \wedge in(p_2,l_2) \wedge in(x,l_3) \wedge in(p_3,l_3))))).\end{aligned}$$

$$(\forall x \forall y(mol(x,y) \equiv\ in(x,y))).$$

$$T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models T_{most\_group}$$

Using Prover9, we have shown that:

$$T_{most\_group} \cup \Delta_2 \models T_{cycle\_path\_subgraph\_nonisolated}$$

$$T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models T_{most\_group}$$

□

Proofs for this theorem can be found in COLORE[11] and in Appendix D.2. Since we have shown that T_{most_group} is synonymous with $T_{cycle\_path\_subgraph\_nonisolated}$, T_{most_group}

[11] http://colore.oor.net/molecular_graph/interprets/

is also consistent.

13.5 Representation Theorem for $Mod(T_{most\_group})$

Up until this point, we have specified a set of ontological commitments based on incidence structures and formalized them as a class of intended structures $\mathfrak{M}^{most\_group}$. We have also proposed a set of axioms for T_{most_group}. In this section, we show that there is a one-to-one correspondence between $\mathfrak{M}^{most\_group}$ and the models of T_{most_group}. The key to this result is the relationship between T_{most_group} and the subtheories that axiomatize the substructures of structures in $\mathfrak{M}^{most\_group}$. From [Grü18], we can utilize the following definition for nonisolated cycle path subgraph incidence structures in the context of points P, lines L, and incidence $\mathbf{I}$:

Definition 13.2: Nonisolated Cycle Path Subgraph Incidence Structures
A nonsiolated cycle path subgraph incidence structure is a tuple $\mathbb{I} = \langle Q \cup L \cup P, \mathbf{plane}, \mathbf{line}, \mathbf{point}, \mathbf{in} \rangle$ *such that*

1. $\mathcal{M} \in \mathfrak{M}^{weak\_tripartite}$;
2. *for each* $\mathbf{l} \in L$,
$$| \, (N(\mathbf{l}) \cap P) \, | = 2$$
3. *for each* $\mathbf{p} \in P$,
$$N(\mathbf{p}) \cap L \neq \emptyset$$
4. *for each* $\mathbf{l} \in L$,
$$1 \leq | \, (N(\mathbf{l}) \cap P) \, | \leq 2$$
5. $Q \subset N(P)$;
6. *for each* $\mathbf{q} \in Q$,
$$(N(N(\mathbf{q}) \cap L) \cap P) = (N(\mathbf{q}) \cap P)$$
7. $P \subset N(Q)$.

$\mathfrak{M}^{cycle\_path\_subgraph}$ *is the class of cycle path subgraph incidence structures.*

From the results in Chapters 13.2 and 13.4, we can state that model of MoSt are synonymous with classes of incidence geometries found in mathematics. Thus, we can state with the following representation theorem:

Theorem 13.5.1: Representation Theorem for T_{most_group}
There exists a bijection

$$\varphi : Mod(T_{most\_group}) \rightarrow \mathfrak{M}^{most\_group}$$

such that $\varphi(\langle \boldsymbol{M}, \boldsymbol{atom}, \boldsymbol{bond}, \boldsymbol{group}, \boldsymbol{mol} \rangle) = \mathbb{I} \oplus \mathbb{B}$ *iff*

1. $M = P \cup L \cup Q$;
2. $\langle \mathbf{a} \rangle \in \mathbf{atom}$ *iff* $\mathbf{a} \in P$;
3. $\langle \mathbf{b} \rangle \in \mathbf{bond}$ *iff* $\mathbf{b} \in L$;
4. $\langle \mathbf{g} \rangle \in \mathbf{group}$ *iff* $\mathbf{g} \in Q$;
5. $\langle \mathbf{x}, \mathbf{y} \rangle \in \mathbf{mol}$ *iff* $\mathbf{y} \in N(\mathbf{x})$.

13.6 Summary

From our verification results, we can see that our axiomatizations of T_{most_graph} and T_{most_group} are synonymous with $T_{nonisolated\_loopless}$ and $T_{cycle\_path\_subgraph\_nonisolated}$. Recall Chapter 3.3 where we presented requirements for the equivalence for models of MoSt and molecules; with these verification results, we have satisfied Requirement (**MOD-1**). Now that we know the ontology is correct with respect to our intended models, Part III discusses applications of the ontology in model construction and reducing the chemical search space.

Part III

Applications of MoSt

Chapter 14

Drug Design as Model Construction

In this chapter, we discuss the application of the ontology in a drug design context. We equate the process of drug design with the process of first-order model construction and discuss the theorems that aid in this composition and design process.

14.1 Revisiting Competency Questions & Requirements

Now that we have all the axioms, as well as verified them against incidence structures found in mathematics, we revisit the competency questions and requirements presented in Chapter 3. With the axioms presented in Part I, we have satisfied Requirements (**R-1**) to (**R-8**) that involved being able to represent the structure of molecules. Additionally, the axioms presented in Part I allow us to write down axioms for molecules that have components that can be connected together.

Relating this back to the discussion of molecules and first-order models of the ontology, we have shown through verification that any molecule is a model of T_{most_root}, and that we can utilize additional axioms as extensions of the ontology to make further restrictions on the types of models we can generate (e.g., developing an extension of the ontology to only allow fused ring molecules, etc.). Further, with these axioms, we can query the knowledge base with molecular description to answer the competency questions pertaining to similarity and substructures.

14.2 Models & Molecules

Here we present a framework that outlines the major concepts in our process of developing models of the ontology:

- A model of MoSt, $Mod(T_{most})$, can be decomposed into 2-connected graph components. Recall from Chapter 11.5.12 that $T_{most} \equiv T_{most\_bridges} \cup T_{most\_ringbond}$, and that T_{most} contains all the attachment and ringbond axioms.

- From the 2-connected components, we can re-compose the graph via the attachment relationships presented in Chapter 7.

- Utilizing representation theorems and satisfiability axioms, we state that models of MoSt and the block decompositions of the underlying molecular graph are synonymous with the tripartite incidence structures presented in the verification of the ontology (see Chapter 12.2).

Figure 14.1 outlines the framework we adopt for the decomposition of molecules into its graph components, and its recomposition into a model of the ontology. We can summarize this figure with the following statements:

- Molecules correspond to models of MoSt.
- Models of MoSt can be constructed by assembling other models of MoSt under certain conditions.
- Molecular descriptions are the axiomatization of the models of MoSt (and hence molecules).
- Using the ontology, we propose new molecules by combining models of MoSt – rather than building models of molecules from scratch using a model builder like Mace4 or Paradox, our strategy is to build and combine models of functional groups with these tools.

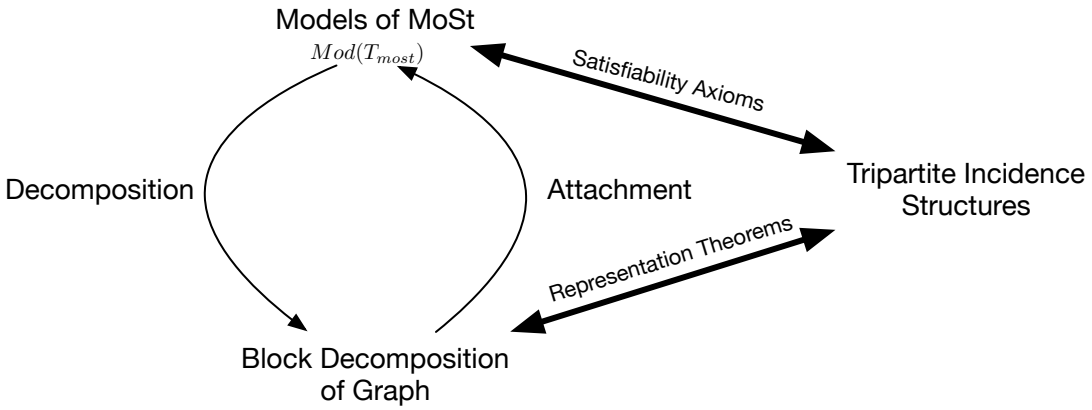
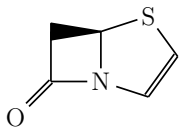


Figure 14.1: Framework for composition and decomposition in MoSt. Solid arrows indicate a single direction, and bolded, solid double arrows indicate bi-directionality (equivalence).

14.3 Molecular Descriptions

Molecular descriptions are considered as definitions in the ontology, where molecules are defined according to their components. For example, Figures 14.2 and 14.3 outline how the ontology is used to define the shape of the penem structure found in penicillins and



(a)

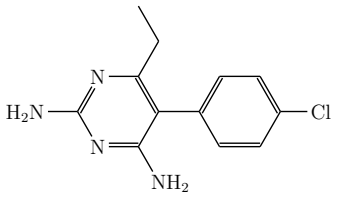
$$\forall x\, penem(x) \equiv \exists g_1 \exists g_2\, skeleton(x) \wedge thiazolidine(g_1) \wedge beta\_lactam(g_2) \wedge mol(g_1, x) \wedge mol(g_2, x) \wedge fused(g_1, g_2)$$

(b) Molecule description for penem.

Figure 14.2: Molecule description for penem (C_5H_5NOS) in the form of a definition.

the daraprim molecule, respectively. Penem (C_5H_5NOS) has a full IUPAC name as 2,3-didehydropenam; we can see in the skeletal formula that it contains a beta lactam ring fused with a thiazolidine core.

Similarly, daraprim ($C_{12}H_{13}CIN_4$) is also known as pyramethamine or 5-(4-chlorophenyl)-6-ethylpyrimidine-2,4-diamine. This drug is used to treat malaria and *Pneumocystis jirovecii* pneumonia (PCP); in 2015, it was subjected to a price hike controversy when Turing Pharmaceuticals raised the price from 13.50 USD to 750 USD per tablet. In the skeletal formula, there is an ethane tethered to the 2,4-Diaminopyrimidine, which is also tethered to the chlorophenyl group of the molecule.



(a)

$$\begin{aligned}\forall x\, pyrimethamine(x) \equiv\ & skeleton(x) \wedge \exists w \exists y \exists z \\ & \exists a_1 \exists a_2 \exists b_1 \exists b_2\, chlorophenyl(w) \wedge \\ & 2,4\_Diaminopyrimidine(y) \wedge \\ & ethane(z) \wedge mol(w, x) \wedge \\ & mol(y, x) \wedge mol(z, x) \wedge \\ & tether(w, y, b_1) \wedge \\ & tether(y, z, b_2)\end{aligned}$$

(b) Molecule description for daraprim.

Figure 14.3: Molecule description for daraprim ($C_{12}H_{13}CIN_4$) in the form of a definition.

As the axioms are succinct enough to outline the structure of molecules, we can utilize these molecular descriptions with model construction.

14.4 Graph Operations for Molecular Graphs

With these molecular descriptions in mind, we discuss the allowable operations for the underlying molecular graphs. These notions are derived from existing graph operations used to produce new graphs from existing graphs. Before we can begin to discuss the graph operations for the molecular graphs, we briefly go over some graph-theoretic concepts for planar graphs. In graph theory, planar graphs are graphs that can be embedded in a plane where no two edges intersect in a point other than at a vertex inside the plane (see Definition 14.1). With MoSt, we axiomatize functional groups as planar graphs.

Definition 14.1: Planar Graph (Adopted from [BR12])
A graph G is planar if there exists a drawing of G in the plane in which no two edges intersect in a point other than a vertex of G, where each edge is a simple arc. In this case, we say that G has been embedded in the plane.

We also introduce the notion of a complete graph (see Definition 14.2) as we refer to a complete graph of four vertices, K_4, later in this work. Complete graphs are depicted in Figure 14.4.

Definition 14.2: Complete Graph (Adopted from [BR12])
A simple graph G is said to be complete if every pair of distinct vertices of G are adjacent in G. Any two complete graphs each on a set of n are isomorphic – each such graph is denoted by K_n.

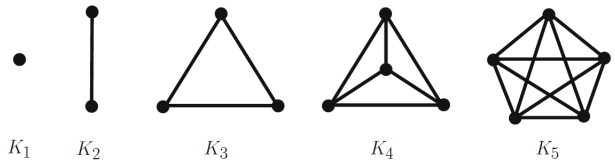
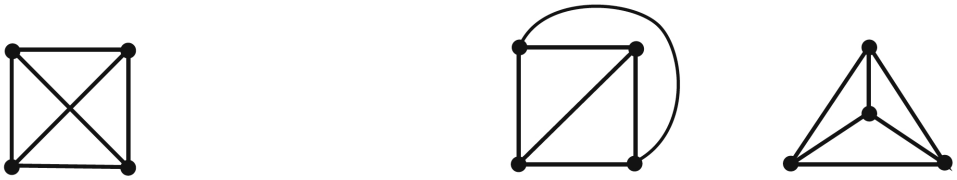


Figure 14.4: Complete graphs with 1, 2, 3, 4, and 5 vertices, denoted as K_1, K_2, K_3, K_4, and K_5, respectively.

In Figure 14.5, we illustrate a complete planar graph for four (4) vertices and its corresponding planar embeddings.



(a) A complete planar graph K_4 **(b)** Planar embeddings for K_4

Figure 14.5: Example of a complete K_4 graph and its planar embeddings. (Figure 8.1 in [BR12])

Recall that we have the following permissible attachments between functional groups, as previously discussed in Chapter 7:

- Spiro – two groups are connected by a shared atom
- Fusion – two groups are connected by sharing one or more bonds
- Tether – two groups are connected by a bond that is not shared between them

With MoSt, we utilize the notion of *path bonding* to combine primitive functional groups together, and map these bonding notions to the permissible attachments for molecules. Recall Definition 12.5, where a path can be defined in terms of the number of its vertices

(e.g., P_1 means a complete graph on K_1, which is a single vertex); suppose we have two graphs G_1 and G_2, we can state the following:

(**ATTACH-1**) The spiro attachment in MoSt corresponds to a P_1 bonding where a vertex is connected to another vertex.

(**ATTACH-2**) The fusion attachment in MoSt corresponds to a P_{2n+1} path bonding where edge(s) of both graphs are shared.

(**ATTACH-3**) The tether attachment in MoSt corresponds to a two-step process of introducing a new path that is not in either graph, and taking one end of the path and bonding it to a vertex in G_1, and then taking the other end of the edge to connect it to a vertex in G_2.

We formally define these graph operations as the attachment theorem for models of MoSt in the next section.

14.5 Structure Theorems for Models of MoSt

With the development of MoSt, we realized that new (structure) theorems were needed to ensure that models of the ontology can be constructed by assembling other models of MoSt under certain conditions. *Structure theorems* are used to specify the models of a theory, an approach taken for mathematical theories such as linear orderings, groups, rings, and modules in [Grü+10]. In this case, we provide representation theorems for the decomposition and composition of primitive functional groups in MoSt, and then specify how models of the ontology can be constructed from such building block structures.

New molecules can be proposed by combining models of the ontology using the following two theorems:

1. A decomposition theorem that takes already-existing molecules and breaks them down into their primitive functional groups, and

2. A set of attachment operations that indicate how primitive functional groups, along with compound groups, can be combined together.

These structure theorems tell us the structure of the models of the ontology, where submodels correspond to primitive functional groups that have been axiomatized in the ontology.

14.5.1 Decomposition Theorem for MoSt

In this theorem, we wish to decompose a molecule (a model of MoSt) into a set of ring structures, chain structures, and bond structures, defined in Definitions 14.3 and 14.5, respectively.

Definition 14.3: Ring Structure
A ring structure is a model of T_{most_group} that contains a unique ring.

Definition 14.4: Chain Structure
A chain structure is a model of T_{most_group} that contains a unique chain.

Definition 14.5: Bond Structure
A bond structure is a model of T_{most_graph} that consists of a bond together with its two atoms.

Further, we revisit the notion of the block decomposition of a graph from Theorem 12.3.1, which states that every graph is a union of its blocks. We consider the ring structures, chain structures, and bond structures of the underlying chemical graph as its blocks. With this in mind, Theorem 14.5.1 states that any model of MoSt can be decomposed into primitive functional groups which consist of ring structures and chain structures, and the leftover bonds that are not part of any primitive functional groups.

Theorem 14.5.1: Decomposition Theorem for MoSt
Suppose $\mathcal{M} \in Mod(T_{most})$, there exists a unique set of:

- *ring structures $R_1, R_2, ..., R_n$,*
- *chain structures $C_1, C_2, ..., C_m$, and*
- *bond structures $B_1, L_2, ..., B_p$.*

such that $(\bigcup R_i) \cup (\bigcup C_j) \cup (\bigcup B_k) = \mathcal{M}$.

Decomposition Theorem Proof. Let $\mathcal{M} \in Mod(T_{most})$.

$\varphi(\mathcal{M})$ has a block decomposition with a set of cycles $\{D_1, ..., D_n\}$, paths $\{E_1, ..., E_p\}$, and graph-bridges $\{F_1, ..., F_q\}$.

$$\varphi(\mathcal{M}) = (D_1 \cup ... \cup D_n) \cup (E_1 \cup ... \cup E_p) \cup (F_1 \cup ... \cup F_q)$$

From our characterization theorem for the models of MoSt in Part II (Theorem 13.3.1), we can state that the inverse of the above statement is a model of MoSt:

$$\mathcal{M} = \varphi^{-1}((D_1 \cup ... \cup D_n) \cup (E_1 \cup ... \cup E_p) \cup (F_1 \cup ... \cup F_q))$$

This is equivalent to the union of the inverse of all the clauses:

$$\begin{aligned}\mathcal{M} = &(\varphi^{-1}(D_1) \cup ... \cup \varphi^{-1}(D_n)) \cup \\ &(\varphi^{-1}(E_1) \cup ... \cup \varphi^{-1}(E_p)) \cup \\ &(\varphi^{-1}(F_1) \cup ... \cup \varphi^{-1}(F_q))\end{aligned}$$

Because we know that cycles, paths, and graph-bridges correspond to ring structures, chain structures, and bond structures, respectively, we can state that:

$$(\varphi^{-1}(D_1) \cup ... \cup \varphi^{-1}(D_n)) = (\mathcal{R}_1 \cup ... \cup \mathcal{R}_n)$$
$$(\varphi^{-1}(E_1) \cup ... \cup \varphi^{-1}(E_p)) = (\mathcal{C}_1 \cup ... \cup \mathcal{C}_p)$$
$$(\varphi^{-1}(F_1) \cup ... \cup \varphi^{-1}(F_q)) = (\mathcal{B}_1 \cup ... \cup \mathcal{B}_q)$$

Therefore, we can state:

$$\begin{aligned}\mathcal{M} =&(\mathcal{R}_1 \cup ... \cup \mathcal{R}_n) \cup \\ &(\mathcal{C}_1 \cup ... \cup \mathcal{C}_p) \cup \\ &(\mathcal{B}_1 \cup ... \cup \mathcal{B}_q)\end{aligned}$$

□

Graphically, this theorem is depicted in Figure 14.6, where daraprim is presented in its skeletal form, underlying graph, and its blocks (ring structures, chain structures, bond structures).

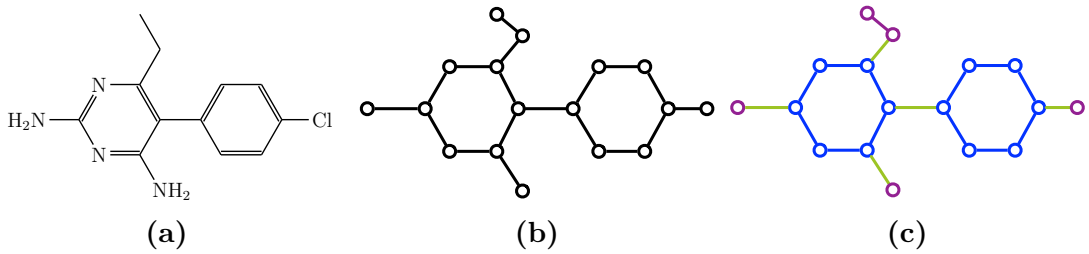


Figure 14.6: Decomposing daraprim ($C_{12}H_{13}ClN_4$). In (a), daraprim is presented in its skeletal form, while the underlying graph for daraprim is presented in (b). In (c), ring structures are outlined in blue, chain structures in purple, and the bond structures in green.

14.5.2 Attachment Theorem for MoSt

Conversely, we also wish to be able to attach models of primitive functional groups together to form new models. Similar to the ontology's attachment relations, we need to limit how these models can be combined together. Before we discuss how we can attach models of MoSt together, we briefly go over the notion of path-bonding, as adopted from [BLS99], in Definition 14.6.

Definition 14.6: Path-Bonding (Adopted from [BLS99])
Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs with $V_1 \cap V_2 = \emptyset$, which both contain paths L_1 and L_2 of size k.

The resulting graph G of the path-bonding operation is obtained by identifying L_1 with L_2, where G has:

$$\textit{vertex set}: (V_1 \cup V_2) \setminus L_2 \textit{ and}$$
$$\textit{edge set}: E_1 \cup E_2 \setminus (\{xy : x, y \in L_2\} \cup \{xy : x \in L_2 \textit{ and } y \in V_2 \setminus L_2\})$$

Given two graphs G and H that contain a complete graph of size k, we can form a new graph in which these graphs are bonded together by identifying the paths [Cor86]. This is graphically depicted in Figure 14.7, where graph J is the result of the K_3 bonding of G and H.

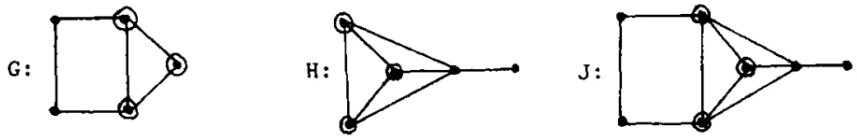


Figure 14.7: Graph J on the right is the result of the K_3 bonding of graphs G and H. (Figure 2 from [Cor86])

With this notion of path bonding in mind, we introduce three attachment operators for models: $\mathfrak{spiro}(\mathcal{M}_1, \mathcal{M}_2)$, $\mathfrak{fusion}(\mathcal{M}_1, \mathcal{M}_2)$, and $\mathfrak{tether}(\mathcal{M}_1, \mathcal{M}_2)$ to represent the spiro, fusion, and tether attachment relations for models of the ontology, respectively. We can state the following for two graphs G_1 and G_2:

1. The spiro attachment, $\mathfrak{spiro}(\mathcal{M}_1, \mathcal{N}_1, \mathcal{M}_2, \mathcal{N}_2)$, is one where we take an existing vertex in model M_1 and connect it together with a vertex in model M_2. $\mathcal{N}_1$ and $\mathcal{N}_2$ denote unique substructures of $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively. Additionally, A_1, A_2, B_1, B_2, G_1, and G_2 denote the atoms, bonds, and functional groups in $\mathcal{M}_1$ and $\mathcal{M}_2$. P_1 signifies a path-graph with 1 vertex of the same type in both $\mathcal{N}_1$ and $\mathcal{N}_2$. Formally,

   > Let $M_1, M_2 \in Mod(T_{most})$, where $\mathcal{M}_1 = \langle A_1 \cup B_1 \cup G_1, mol \rangle$ and $\mathcal{M}_2 = \langle A_2 \cup B_2 \cup G_2, mol \rangle$. The sets of atoms in $\mathcal{M}_1$ and $\mathcal{M}_2$ intersect at one atom x, $A_1 \cap A_2 = \langle x : x \in A_1 \text{ and } x \in A_2 \rangle$. The set of atoms is $A = A_1 \cup A_2$ and the set of bonds is $B = B_1 \cup B_2$.
   >
   > With this, we define $\mathfrak{spiro}$ as follows:
   >
   > $$\mathfrak{spiro}(\mathcal{M}_1, \mathcal{N}_1, \mathcal{M}_2, \mathcal{N}_2) = \mathcal{M}, \text{ where:}$$
   > $$\mathcal{N}_1 \cong P_1 (\text{path graph})$$
   > $$\mathcal{N}_1 \cong \mathcal{N}_2 (\text{isomorphic})$$
   > $$\mathcal{N}_1 \subseteq \mathcal{M}_1 \text{ (substructure)}$$
   > $$\mathcal{N}_2 \subseteq \mathcal{M}_2 \text{ (substructure)}$$

   Graphically, we depict this attachment operator in Figure 14.8. With the operator, we specify the two graphs we would like to attach (R_1 and R_2), as well as the vertices for the attachment ($v_{1,3}$ and $v_{2,5}$). The resulting graph is a combination of R_1 and R_2 connected at the indicated vertices.

2. Similarly, the fusion attachment, $\mathfrak{fusion}(\mathcal{M}_1, \mathcal{N}_1, \mathcal{M}_2, \mathcal{N}_2)$, is one where we take an existing bond in model M_1 and fuse it together with a bond in model M_2. $\mathcal{N}_1$ and $\mathcal{N}_2$ denote unique substructures of $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively. Additionally, A_1, A_2,

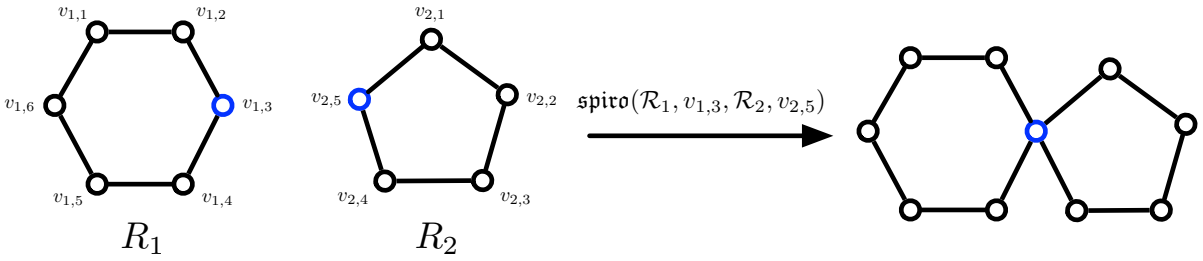


Figure 14.8: Example attachment operation for $\mathfrak{spiro}(\mathcal{R}_1, v_{1,3}, \mathcal{R}_2, v_{2,5})$, where R_1 and R_2 are rings, and $v_{1,3}$ and $v_{2,5}$ are vertices in R_1 and R_2, respectively. The vertices are denoted in blue in R_1 and R_2, and the attachment location is also denoted in blue in the resulting graph.

B_1, B_2, G_1, and G_2 denote the atoms, bonds, and functional groups in $\mathcal{M}_1$ and $\mathcal{M}_2$. P_{2n+1} signifies a path-graph with n vertices.

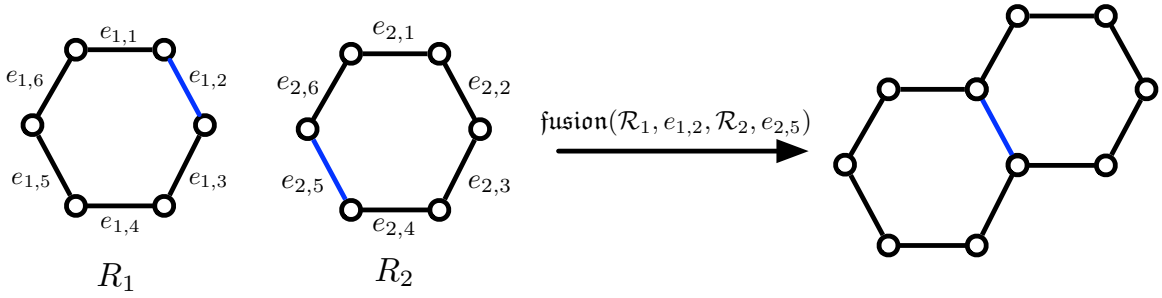
Let $M_1, M_2 \in Mod(T_{most})$, where $\mathcal{M}_1 = \langle A_1 \cup B_1 \cup G_1, mol \rangle$ and $\mathcal{M}_2 = \langle A_2 \cup B_2 \cup G_2, mol \rangle$. The sets of atoms in $\mathcal{M}_1$ and $\mathcal{M}_2$ do not intersect: $A_1 \cap A_2 = \emptyset$. The set of atoms is $A = (A_1 \cup A_2) \setminus \mathcal{N}_2$ and the set of bonds is $B = B_1 \cup B_2 \setminus (\{xy : x, y \in \mathcal{N}_2\} \cup \{xy : x \in \mathcal{N}_2 \text{ and } y \in A_2 \setminus \mathcal{N}_2\})$. With this, we define $\mathfrak{fusion}$ as follows:

$$\mathfrak{fusion}(\mathcal{M}_1, \mathcal{N}_1, \mathcal{M}_2, \mathcal{N}_2) = \mathcal{M}, \text{ where:}$$
$$\mathcal{N}_1 \cong P_{2n+1}, n \geq 1 \text{ (path graph)}$$
$$\mathcal{N}_1 \cong \mathcal{N}_2 \text{ (isomorphic)}$$
$$\mathcal{N}_1 \subseteq \mathcal{M}_1 \text{ (substructure)}$$
$$\mathcal{N}_2 \subseteq \mathcal{M}_2 \text{ (substructure)}$$

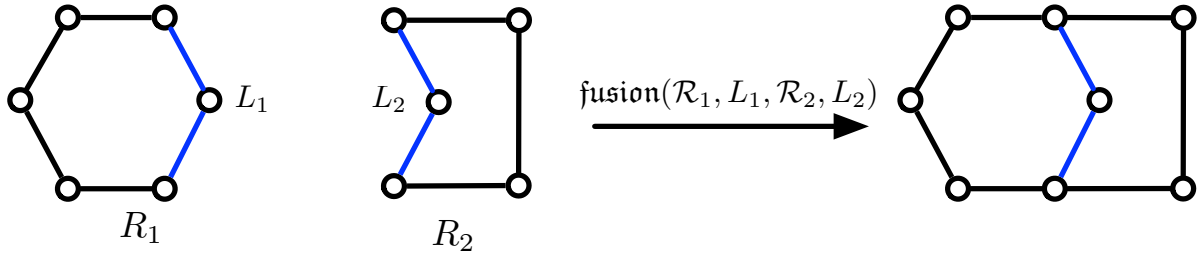
Graphically, we depict this attachment operator in Figure 14.9. In both cases, we specify the two graphs we would like to attach (R_1 and R_2), as well as the path for the attachment ($v_{1,3}$ and $v_{2,5}$). The resulting graph is a combination of R_1 and R_2 connected at the indicated edges/paths. Figure 14.9(a) illustrates fusion along a unique edge in R_1 and R_2, whereas Figure 14.9(b) illustrates fusion along paths of the same size in R_1 and R_2.

3. Finally, the tether attachment, $\mathfrak{tether}(M_1, M_2)$, is a two-step process of taking a complete graph on 2 vertices (K_2) that is not in G_1 nor in G_2 and using the spiro attachment to connect it to a vertex in G_1 and connect to another vertex in G_2. We can define it as such:

   Let $M_1, M_2, K_2 \in Mod(T_{most})$, where $\mathcal{M}_1 = \langle A_1 \cup B_1 \cup G_1, mol \rangle$ and $\mathcal{M}_2 = \langle A_2 \cup B_2 \cup G_2, mol \rangle$. K_2 is a complete graph on 2 (a path). The sets of atoms in $\mathcal{M}_1$ and $\mathcal{M}_2$ do not intersect: $A_1 \cap A_2 = \emptyset$. The set of atoms is $A = A_1 \cup A_2$ and the set of bonds is $B = B_1 \cup B_2$. With this,



(a) Fusing two rings along one edge using the attachment operation for $\mathfrak{fusion}(\mathcal{R}_1, e_{1,2}, \mathcal{R}_2, e_{2,5})$, where R_1 and R_2 are rings, and $e_{1,2}$ and $e_{2,5}$ are edges in R_1 and R_2, respectively. The edges are denoted in blue in R_1 and R_2, and the attachment location is also denoted in blue in the resulting graph.



(b) Fusing two rings along a path using the attachment operation for $\mathfrak{fusion}(\mathcal{R}_1, L_1, \mathcal{R}_2, L_2)$, where R_1 and R_2 are rings, and L_1 and L_2 are paths in R_1 and R_2, respectively. The paths are denoted in blue in R_1 and R_2, and the attachment location is also denoted in blue in the resulting graph.

Figure 14.9: Example attachment operation for $\mathfrak{fusion}(\mathcal{R}_1, e_{1,2}, \mathcal{R}_2, e_{2,5})$ and $\mathfrak{fusion}(\mathcal{R}_1, L_1, \mathcal{R}_2, L_2)$. In (a), fusion occurs along one edge in R_1 and R_2, whereas fusion in (b) occurs along paths L_1 and L_2 in R_1 and R_2. The vertices are denoted in blue in R_1 and R_2, and the attachment location is also denoted in blue in the resulting graph.

we define $\mathfrak{tether}$ as a composite operator as follows:

$$\mathfrak{tether}(\mathcal{M}_1, \mathcal{N}_1, \mathcal{M}_2, \mathcal{N}_2) = \mathfrak{spiro}(\mathcal{M}_1, \mathcal{N}_1, \mathfrak{spiro}(K_2, K_2, \mathcal{M}_2, \mathcal{N}_2), \mathcal{N}_2), \text{ where:}$$
$$K_2 = \{xy : x \in \mathcal{N}_1 \text{ and } y \in \mathcal{N}_2\} \text{ (path graph)}$$
$$\mathcal{N}_1 \subseteq \mathcal{M}_1 \text{ (substructure)}$$
$$\mathcal{N}_2 \subseteq \mathcal{M}_2 \text{ (substructure)}$$

In Theorem 14.5.2, we specify the attachment theorem for models of MoSt: models can be combined and created using the $\mathfrak{spiro}(\mathcal{M}_1, \mathcal{N}_1, \mathcal{M}_2, \mathcal{N}_2)$, $\mathfrak{fusion}(\mathcal{M}_1, \mathcal{N}_1, \mathcal{M}_2, \mathcal{N}_2)$, and $\mathfrak{tether}(\mathcal{M}_1, \mathcal{N}_1, \mathcal{M}_2, \mathcal{N}_2)$ attachment operators.

Theorem 14.5.2: Attachment Theorem for MoSt
$Mod(T_{most})$ is closed under the following attachment operations:

1. $\mathfrak{spiro}(\mathcal{M}_1, \mathcal{N}_1, \mathcal{M}_2, \mathcal{N}_2) \in Mod(T_{most})$ *iff* $\mathcal{M}_1, \mathcal{M}_2 \in Mod(T_{most})$
2. $\mathfrak{fusion}(\mathcal{M}_1, \mathcal{N}_1, \mathcal{M}_2, \mathcal{N}_2) \in Mod(T_{most})$ *iff* $\mathcal{M}_1, \mathcal{M}_2 \in Mod(T_{most})$
3. $\mathfrak{tether}(\mathcal{M}_1, \mathcal{N}_1, \mathcal{M}_2, \mathcal{N}_2) \in Mod(T_{most})$ *iff* $\mathcal{M}_1, \mathcal{M}_2 \in Mod(T_{most})$

With this theorem, we can state that additional models of T_{most} can be composed of models of T_{most} using the attachment operators.

14.6 Relationship Between Graph Operators & Model Operators

Now that we have theorems for the decomposition of models and attachment of models of MoSt, we have sets of attachment operators for the models that need to be mapped with graph-theoretic operators for attachment (path-bonding operators). With Theorem 14.5.1, we can state the following:

> Let γ^n be a path-bonding operation on two graphs G_1 and G_2, where n signifies the number of vertices involved in the path-bond.
>
> Let $\mathcal{M} \in Mod(T_{most})$. $\varphi(\mathcal{M})$ has a block decomposition with a set of cycles $\{D_1, ..., D_n\}$, paths $\{E_1, ..., E_p\}$, and graph-bridges $\{F_1, ..., F_q\}$. We can map the graph operations with operations on the models of the ontology:
>
> - $\varphi(\mathfrak{spiro}(\mathcal{M}_1, \mathcal{N}_1, \mathcal{M}_2, \mathcal{N}_2)) = \gamma^1(\varphi(\mathcal{M}_1), \varphi(\mathcal{M}_2))$
> - $\varphi(\mathfrak{fusion}(\mathcal{M}_1, \mathcal{N}_1, \mathcal{M}_2, \mathcal{N}_2)) = \gamma^{|\mathcal{N}_1|}(\varphi(\mathcal{M}_1), \varphi(\mathcal{M}_2))$, where $|\mathcal{N}_1|$ signifies the cardinality of $\mathcal{N}_1$.
> - $\varphi(\mathfrak{tether}(\mathcal{M}_1, \mathcal{N}_1, \mathcal{M}_2, \mathcal{N}_2)) = \gamma^1(\gamma^1(\varphi(\mathcal{M}_1), K_2), \varphi(\mathcal{M}_2))$ where K_2 is a complete graph on 2.

Each of these mappings are lemmas and are proven in turn below.

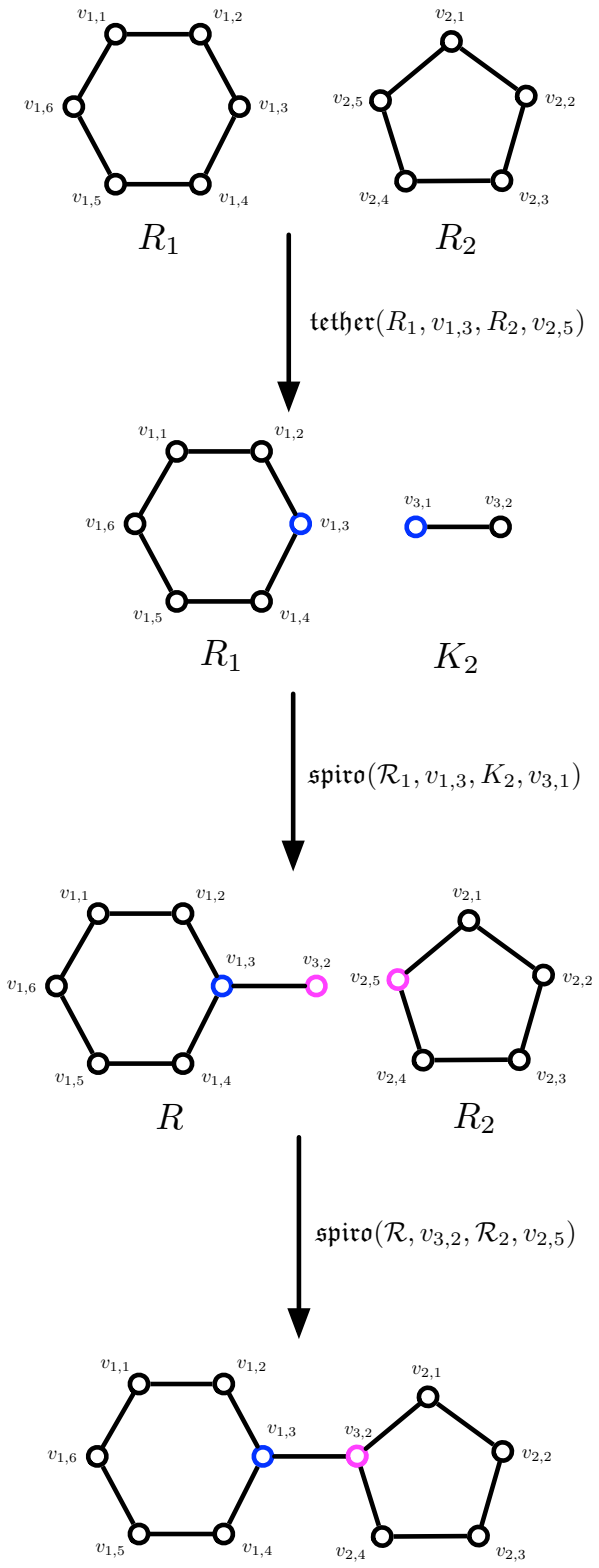


Figure 14.10: Example attachment operation for $\mathfrak{tether}(\mathcal{R}_1, v_{1,3}, \mathcal{R}_2, v_{2,5})$, where R_1 and R_2 are rings, and $v_{1,3}$ and $v_{2,5}$ are vertices in R_1 and R_2, respectively. Additionally, we introduce a simple path graph with one edge, labelled K_2, which will be attached to both rings in separate $\mathfrak{spiro}$ operations. In the first $\mathfrak{spiro}$ operation, the vertices are denoted in blue in R_1 and K_2, and then denoted in purple in the second $\mathfrak{spiro}$ operation in R and R_2.

Lemma 14.6.1: Spiro Mapping
The $\mathfrak{spiro}$ *operator on models can be mapped to a corresponding operation on the graphs associated with the models.*

$$\varphi(\mathfrak{spiro}(\mathcal{M}_1, \mathcal{N}_1, \mathcal{M}_2, \mathcal{N}_2)) = \gamma^1(\varphi(\mathcal{M}_1), \varphi(\mathcal{M}_2))$$

Spiro Mapping Proof. Let G_1 and G_2 be two graphs with models $\mathcal{M}_1$ and $\mathcal{M}_2$. Let R, L, Q, I denote point, line, plane, and incidence.

Using the definition of the $\mathfrak{spiro}$ operator, we can state:

$$\varphi(\mathcal{M}_1) = G_1 \quad G_1 = \langle R_1, L_1, Q_1, I_1 \rangle$$
$$\varphi(\mathcal{M}_2) = G_2 \quad G_2 = \langle R_2, L_2, Q_2, I_2 \rangle$$

$$\varphi(A_1 \cap A_2) = \varphi(A_1) \cap \varphi(A_2) = R_1 \cap R_2$$
$$|A_1 \cap A_2| = 1 \iff |R_1 \cap R_2| = 1$$

$$\varphi(B_1 \cup B_2) = \varphi(B_1) \cup \varphi(B_2) = L_1 \cup L_2$$

$$\varphi(\mathfrak{spiro}(\mathcal{M}_1, \mathcal{N}_1, \mathcal{M}_2, \mathcal{N}_2)) = \gamma^1(G_1, G_2)$$

□

Let $\mathcal{M}_1, \mathcal{M}_2 \in Mod(T_{most})$. By Lemma 14.6.1, the spiro of two models is the result of the spiro of two graphs, where $\gamma^{-1}(\varphi(\mathcal{M}_1), \varphi(\mathcal{M}_2))$ is the image of the model. We can state that the inverse of this mapping is a model of T_{most}:

$$\varphi^{-1}(\gamma^{-1}(\varphi(\mathcal{M}_1), \varphi(\mathcal{M}_2))) \in Mod(T_{most})$$

Lemma 14.6.2: Fusion Mapping
The $\mathfrak{fusion}$ *operator on models can be mapped to a corresponding operation on the graphs associated with the models.*

$$\varphi(\mathfrak{fusion}(\mathcal{M}_1, \mathcal{N}_1, \mathcal{M}_2, \mathcal{N}_2)) = \gamma^{|\mathcal{N}_1|}(\varphi(\mathcal{M}_1), \varphi(\mathcal{M}_2))$$

Fusion Mapping Proof. Let G_1 and G_2 be two graphs with models $\mathcal{M}_1$ and $\mathcal{M}_2$. Let R, L, Q, I denote point, line, plane, and incidence.

Using the definition of the $\mathfrak{fusion}$ operator, we can state:

$$\varphi(\mathcal{M}_1) = G_1 \quad G_1 = \langle R_1, L_1, Q_1, I_1 \rangle$$
$$\varphi(\mathcal{M}_2) = G_2 \quad G_2 = \langle R_2, L_2, Q_2, I_2 \rangle$$

$$\varphi(\mathcal{N}_1) = H_1 \quad H_1 \subseteq G_1$$
$$\varphi(\mathcal{N}_2) = H_2 \quad H_2 \subseteq G_2$$

$$\varphi(A_1 \cap A_2) = \varphi(A_1) \cap \varphi(A_2) = R_1 \cap R_2$$
$$|A_1 \cap A_2| = 0 \iff |R_1 \cap R_2| = 0$$

$$\varphi(A_1 \cup A_2 \setminus \mathcal{N}_1) = \varphi(A_1) \cup \varphi(A_2) \setminus \varphi(\mathcal{N}_1) = R_1 \cup R_2 \setminus H_1$$

$$\varphi(B_1 \cup B_2) = \varphi(B_1) \cup \varphi(B_2) = L_1 \cup L_2$$

$$\varphi(\mathfrak{fusion}(\mathcal{M}_1, \mathcal{N}_1, \mathcal{M}_2, \mathcal{N}_2)) = \gamma^{|\mathcal{N}_1|}(G_1, G_2)$$

□

Let $\mathcal{M}_1, \mathcal{M}_2 \in Mod(T_{most})$. By Lemma 14.6.2, the fusion of two models is the result of the fusion of two graphs, where $\gamma^{-|\mathcal{N}_1|}(\varphi(\mathcal{M}_1), \varphi(\mathcal{M}_2))$ is the image of the model. We can state that the inverse of this mapping is a model of T_{most}:

$$\varphi^{-1}(\gamma^{-|\mathcal{N}_1|}(\varphi(\mathcal{M}_1), \varphi(\mathcal{M}_2))) \in Mod(T_{most})$$

Lemma 14.6.3: Tether Mapping
The $\mathfrak{tether}$ *operator on models can be mapped to a corresponding operation on the graphs associated with the models.*

$$\varphi(\mathfrak{tether}(\mathcal{M}_1, \mathcal{N}_1, \mathcal{M}_2, \mathcal{N}_2)) = \gamma^1(\gamma^1(\varphi(\mathcal{M}_1), K_2), \varphi(\mathcal{M}_2))$$

Tether Mapping Proof. Let G_1 and G_2 be two graphs with models $\mathcal{M}_1$ and $\mathcal{M}_2$. Let R, L, Q, I denote point, line, plane, and incidence.

Using the definition of the $\mathfrak{tether}$ operator, we can state:

$$\varphi(\mathcal{M}_1) = G_1 \quad G_1 = \langle R_1, L_1, Q_1, I_1 \rangle$$
$$\varphi(\mathcal{M}_2) = G_2 \quad G_2 = \langle R_2, L_2, Q_2, I_2 \rangle$$

$$\varphi(A_1 \cap A_2) = \varphi(A_1) \cap \varphi(A_2) = R_1 \cap R_2$$
$$|A_1 \cap A_2| = 0 \iff |R_1 \cap R_2| = 0$$

$$\varphi(A_1 \cup A_2) = \varphi(A_1) \cup \varphi(A_2)$$
$$\varphi(B_1 \cup B_2) = \varphi(B_1) \cup \varphi(B_2) = L_1 \cup L_2$$

$$\varphi(A_1 \cap K_2) = \varphi(A_1) \cap \varphi(K_2) = R_1 \cap K_2$$
$$|A_1 \cap K_2| = 1 \iff |R_1 \cap K_2| = 1$$

$$\varphi(A_2 \cap K_2) = \varphi(A_1) \cap \varphi(K_2) = R_2 \cap K_2$$
$$|A_2 \cap K_2| = 1 \iff |R_2 \cap K_2| = 1$$

$$\varphi(\mathfrak{tether}(\mathcal{M}_1, \mathcal{N}_1, \mathcal{M}_2, \mathcal{N}_2)) = \gamma^1(\gamma^1(G_1, K_2), G_2)$$

□

Let $\mathcal{M}_1, \mathcal{M}_2 \in Mod(T_{most})$. By Lemma 14.6.3, the tether of two models is the result of the tether of two graphs, where $\gamma^{-1}(\gamma^{-1}(\varphi(\mathcal{M}_1), K_2), \varphi(\mathcal{M}_2))$ is the image of the model. We can state that the inverse of this mapping is a model of T_{most}:

$$\varphi^{-1}(\gamma^{-1}(\gamma^{-1}(\varphi(\mathcal{M}_1), K_2), \varphi(\mathcal{M}_2))) \in Mod(T_{most})$$

From these lemmas, we also introduce the following statements:

1. Attachment does not create or destroy any rings:
   - R is a ring structure in $\mathfrak{spiro}(\mathcal{M}_1, \mathcal{N}_1, \mathcal{M}_2, \mathcal{N}_2)$ iff R is a ring structure in $\mathcal{M}_1$ and $\mathcal{M}_2$.
   - R is a ring structure in $\mathfrak{fusion}(\mathcal{M}_1, \mathcal{N}_1, \mathcal{M}_2, \mathcal{N}_2)$ iff R is a ring structure in $\mathcal{M}_1$ and $\mathcal{M}_2$.
   - R is a ring structure in $\mathfrak{tether}(\mathcal{M}_1, \mathcal{N}_1, \mathcal{M}_2, \mathcal{N}_2)$ iff R is a ring structure in $\mathcal{M}_1$ and $\mathcal{M}_2$.
2. Attachment does not create or destroy any chains:
   - C is a chain structure in $\mathfrak{spiro}(\mathcal{M}_1, \mathcal{N}_1, \mathcal{M}_2, \mathcal{N}_2)$ iff C is a chain structure in $\mathcal{M}_1$ and $\mathcal{M}_2$.
   - C is a chain structure in $\mathfrak{tether}(\mathcal{M}_1, \mathcal{N}_1, \mathcal{M}_2, \mathcal{N}_2)$ iff C is a chain structure in $\mathcal{M}_1$ and $\mathcal{M}_2$.

Now that we have proved Theorems 14.5.1 and 14.5.2, we discuss procedures for generating models via the attachment axioms of MoSt.

14.7 Generating Models via Attachment

With the attachment operators in mind, we can develop two methods to generate models with the ontology:

Method #1: We can utilize the mathematical notion of chordal graphs and use these to build *attachment graphs*.

Method #2: We can use a recursive method of generating models using a model builder such as Mace4.

We describe each of these below.

14.7.1 Method #1: Building Attachment Graphs for MoSt

We have devised an alternative approach to looking at attachment between primitive functional groups in the ontology by utilizing chordal graphs to build what we call *attachment graphs* for model generation. We describe each of these in turn below and outline the procedure we developed to build models of MoSt.

Chordal Graphs

Consider the notion of chordal graphs in graph theory, as defined in [BLS99] in Definition 14.7. Chordal graphs represent a class of graphs that have nice cutset properties, and are also referred to as triangulated graphs, rigid-circuit graphs, monotone transitive graphs, and perfect elimination graphs in mathematics [BLS99].

> **Definition 14.7: Chordal Graph (Adopted from [BLS99])**
> *A graph G is chordal if each cycle in G of length at least 4 has at least one chord. In other words, a chordal graph is a graph possessing no chordless cycles, and every chordal graph is perfect.*

In simpler terms, an undirected graph is considered *chordal* if every cycle (of length greater than three) possesses an edge joining two nonconsecutive vertices of the cycle ('a chord'). Figure 14.11 outlines some examples of chordal graphs, where chords are indicated in blue – if removed, the graphs become non-chordal.

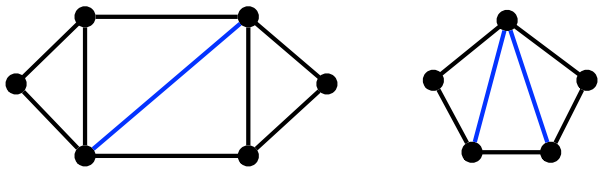


Figure 14.11: Examples of chordal graphs. The blue lines indicate chords in the graphs. If removed, the graphs then become non-chordal.

Attachment Graphs

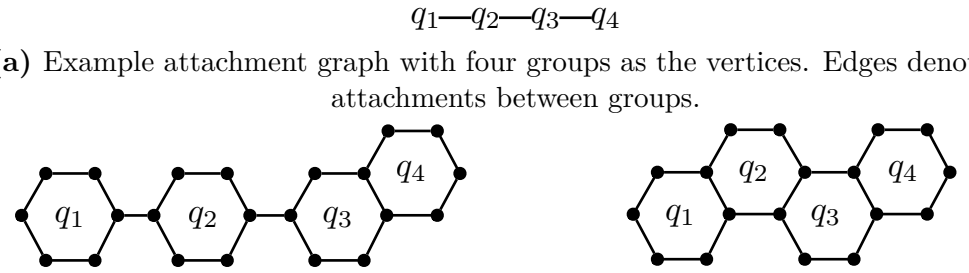
Using this definition of chordal graph, we introduce our notion of an *attachment graph*. Since we know that primitive functional groups are rings and chains, we can then state that it is possible for us to generate attachment graphs that provide a flexible method of attaching groups together. As a meta-level schematic, *attachment graphs* permit us to easily substitute primitive functional groups and attachment types (see Definition 14.8). For example, Figure 14.12 illustrates how attachment graphs are *not* unique, as there are

multiple permutations of arrangements possible for the functional groups and attachment operators.

Definition 14.8: Attachment Graph
An attachment graph is a meta-level schematic of indicating how functional groups are attached together in a molecule. It is a high-level view of the models of the ontology: vertices of the attachment graph are primitive functional groups and the edges are the types of attachments between the groups.

For attachment graphs that have cycles, the attachment type in the cycle must be fusion. See Figure 14.14.

Attachment graphs are not unique, as they can be constructed by substituting the vertices of the attachment graph with primitive functional groups (rings and chains). See Figure 14.12.



(a) Example attachment graph with four groups as the vertices. Edges denote the attachments between groups.

(b) Arrangement of groups and attachments.

(c) Alternative arrangement of groups and attachments.

Figure 14.12: Example of an attachment graph.

Another example of an attachment graph is shown in Figure 14.13, where q_1 and q_2 can be rings or chains. Since we allow the flexibility for the attachment graph to not be unique, a simple attachment graph such as the one shown in Figure 14.13(a) allows for multiple possibilities. For example, the attachment graph allows for the following possible variations (not exhaustive):

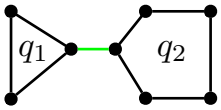
- Figure 14.13(b) outlines two rings q_1 and q_2 that are fused together,
- Figure 14.13(c) outlines two rings q_1 and q_2 that are tethered together (by a bond highlighted in green), and
- Figure 14.13(d) outlines a ring q_1 that is connected to a chain q_2 via the spiro attachment (the chain is denoted in blue).

With attachment graphs that contain cycles, the attachment relationship *must* be fusion. In order for the resultant graph (with functional groups) to be chordal, we strictly enforce that **cyclic attachment graphs only contain the fusion attachment**, as shown in Figures 14.14.

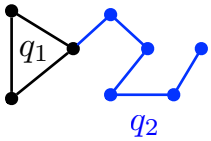
A more complex attachment graph is shown in Figure 14.15, where we have a cycle between q_1, q_2 and q_3 that is connected to q_4 and q_5. Because attachment graphs are

(a) Example attachment graph with two groups as the vertices. Edges denote the attachments between groups.

(b) Fusion arrangement between q_1 and q_2.

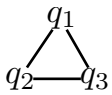


(c) Tether arrangement between q_1 and q_2.

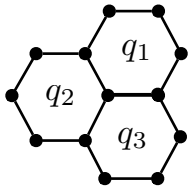


(d) Spiro arrangement between q_1 and q_2, where q_2 is denoted in blue for readability purposes.

Figure 14.13: Another example of an attachment graph.



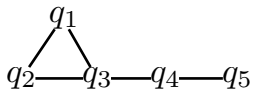
(a) Example attachment graph with three groups as the vertices in a cycle. Edges denote the attachments between groups.



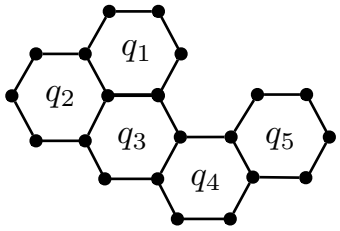
(b) Corresponding fusion arrangement between q_1, q_2, and q_3.

Figure 14.14: For attachment graphs that contain cycles, the attachment relationship must be fusion.

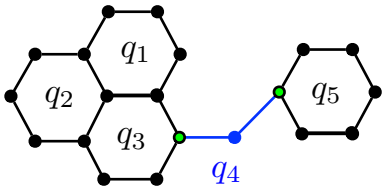
not unique, we have various possibilities for these connections between all the primitive functional groups.



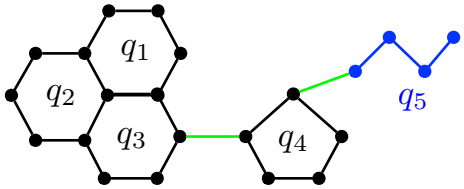
(a) Example attachment graph with five groups as the vertices. Edges denote the attachments between groups.



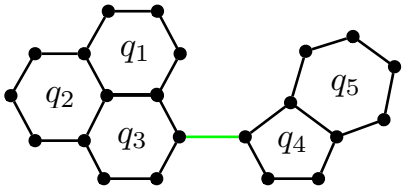
(b) All groups are fused together.



(c) q_1, q_2, and q_3 are fused, while q_4 (highlighted in blue) is connected to q_3 and q_5 via the spiro attachment. The green atoms are the spiro atoms.



(d) q_1, q_2, and q_3 are fused, while q_4 is tethered to q_3 and q_5. The green bonds indicate the tether bonds.



(e) q_1, q_2, and q_3 are fused, while q_4 is fused with q_5 while being tethered to q_3. The green bonds indicate the tether bonds.

Figure 14.15: A more complex attachment graph is shown in (a), with various combinations of attachments between the groups shown in (b) to (e) (not exhaustive). Since q_1, q_2, and q_3 are in a cycle in (a), this means that the groups must be fused in the resultant graphs.

Recall Figure 10.5 from Chapter 10.2, where we had the breakdown of ethyl acetate into its various skeletons and functional groups. Recall how the breakdown of skeletons and groups showed the connections between groups – these connections are essentially the attachment graphs described in this section, outlined in Figure 14.16.

Procedure 14.1: Generating Chordal Graphs

In [Neb07], the author presents a *pseudo-elementary axiomatization* for the class of chordal graphs; as chordal graphs are not finitely axiomatizable [BRW85; AL95], Nebeský presents a finite first-order axiomatization in the expanded signature of graphs and ternary systems which we will use to assist us with building models of MoSt. From what we've seen in Part II, we want to leverage the usage of theorem provers in this work – we are interested in the axioms of [Neb07] as it provides a basis for us to formulate a theorem proving problem for chordal graphs. In particular, we are interested in Ax-

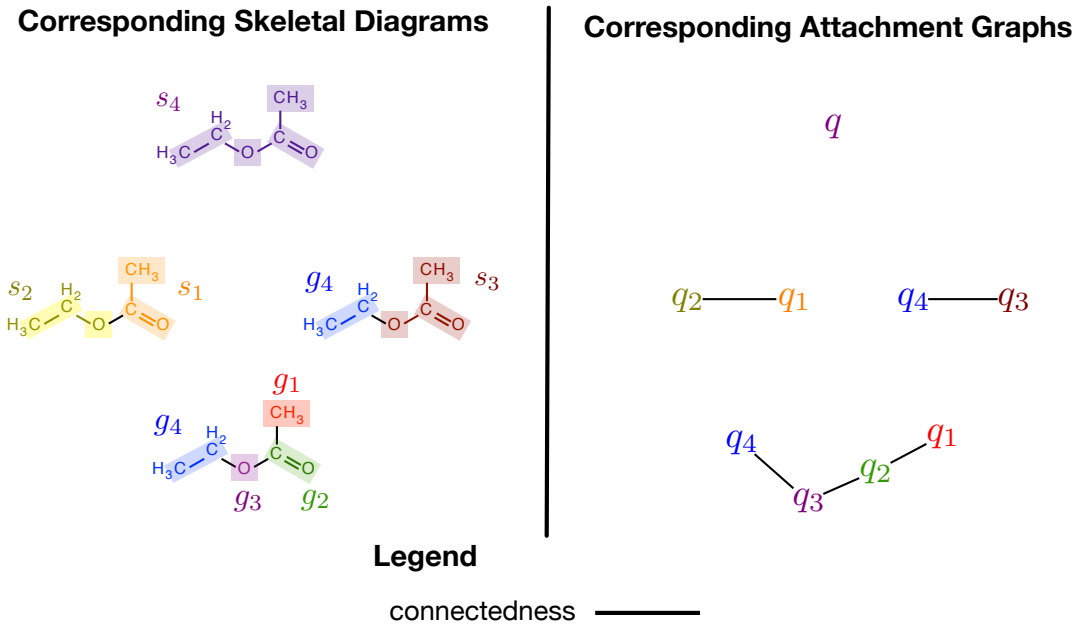


Figure 14.16: Comparison between attachment graph and breakdown of skeletons into their functional groups. Recall Figure 10.5 which outlines the various compositions of the skeletons for ethyl acetate – we have reproduced this here and have drawn out the corresponding attachment graphs for these skeletal diagrams on the right hand side.

ioms (A1) to (A5) in [Neb07], which we have included in Figure 14.17 as a theory called $T_{nebesky\_ip\_connected\_chordal}$[1] (and have re-labelled as Axioms NC-A1 to NC-A5, respectively). Nebeský presents the notions of an induced-path system, denoted as *ip-system*, as a component of a ternary system S of a graph G.

Nebeský indicates that, with Axioms NC-A1 to NC-A5, a ternary system that satisfies Axioms NC-A1 and NC-A2 with an underlying graph G also results in the fact that G is a connected chordal graph with S as the ip-system of G if it satisfies Axioms NC-A3, NC-A4, and NC-A5 (this proposition is quoted below in Figure 14.18).

To generate chordal graphs, we have Procedure 14.1 that calls Mace4 to generate non-trivial models of the ontology. This procedure will also be used in conjunction with other procedures outlined in Chapter 14.7.2. Examples of chordal graphs generated by Mace4 using Nebeský's axioms are shown in Figure 14.19. The elements of the models are also listed for reference. With Mace4, we also include the following axiom to indicate that an edge of a graph is mapped to Nebeský's ternary relation: $(\forall x \forall y(E(x,y) \equiv S(x,y,y)))$. In Figure 14.19, we use $E(x,y)$ to signify the edges of the chordal graph that are part of the model M.

With Procedure 14.1, this allows us to generate chordal graphs which we then use as attachment graphs to build models of the ontologies by substituting the vertices of the chordal graphs with functional groups. We are extending these attachment graphs to construct new models of MoSt using theorem provers. As arbitrary chordal graphs are

[1] `http://colore.oor.net/nebesky_ternary/ip_connected_chordal.clif`

$$(\forall u \forall v \forall w\ (S(u,v,w) \supset S(v,u,u))). \tag{NC-A1}$$

$$(\forall u \forall v \forall w\ (S(u,v,w) \supset w \neq u \wedge u \neq v \wedge w \neq v)). \tag{NC-A2}$$

$$(\forall u \forall v \forall w \forall x\ (S(u,v,v) \wedge S(v,w,x) \wedge u \neq w \wedge \neg S(u,w,w) \supset S(u,v,x))). \tag{NC-A3}$$

$$(\forall u \forall v \forall w\ (S(u,v,w) \wedge v \neq w \supset (\exists y(S(v,y,w) \wedge y \neq u \wedge \neg S(u,y,y))))). \tag{NC-A4}$$

$$(\forall u \forall v\ (u \neq v \supset (\exists z S(u,z,v)))). \tag{NC-A5}$$

Figure 14.17: Axioms (A1) to (A5) from [Neb07], which we have renumbered and grouped as a theory called $T_{nebesky\_ip\_connected\_chordal}$ in COLORE.

Let G be a graph, and let S denote the ip-system of G. Then,

a S satisfies NC-A1, NC-A2, and NC-A4, and G is the underlying graph of S;

b if G is connected, then S satisfies axiom NC-A5;

c if G is chordal, then S satisfies the axiom NC-A3.

Figure 14.18: Proposition #1 in [Neb07]

Procedure 14.1 chordalgraph

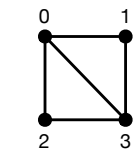
Input: Chordal Graph Axioms NC-A1 to NC-A5
Output: Mace4 Model $\mathcal{M} \in Mod(T_{nebesky\_ip\_connected\_chordal})$

1: **procedure** CHORDALGRAPH
2: Generate non-trivial model ▷ Using Mace4

$$\mathcal{M} \in Mod(T_{nebesky\_ip\_connected\_chordal})$$

3: **return** $\mathcal{M}$ ▷ Return the model
4: **end procedure**

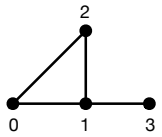
built using Procedure 14.1, we are not concerned with the computation time required for these chordal graphs.



(a)

$\mathbf{M} = \{0, 1, 2, 3\}$
$\mathbf{E} = \{\langle 0,1\rangle, \langle 0,2\rangle, \langle 0,3\rangle, \langle 1,0\rangle, \langle 1,3\rangle, \langle 2,0\rangle, \langle 2,3\rangle, \langle 3,0\rangle, \langle 3,1\rangle, \langle 3,2\rangle\}$

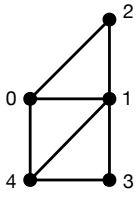
(b) Corresponding model for (a).



(c)

$\mathbf{M} = \{0, 1, 2, 3\}$
$\mathbf{E} = \{\langle 0,1\rangle, \langle 0,2\rangle, \langle 1,0\rangle, \langle 1,2\rangle, \langle 1,3\rangle, \langle 2,0\rangle, \langle 2,1\rangle, \langle 3,1\rangle\}$

(d) Corresponding model for (c).



(e)

$\mathbf{M} = \{0, 1, 2, 3, 4\}$
$\mathbf{E} = \{\langle 0,1\rangle, \langle 0,2\rangle, \langle 0,4\rangle, \langle 1,0\rangle, \langle 1,2\rangle, \langle 1,3\rangle, \langle 1,4\rangle, \langle 2,0\rangle, \langle 2,1\rangle, \langle 3,1\rangle, \langle 3,4\rangle, \langle 4,0\rangle, \langle 4,1\rangle, \langle 4,3\rangle\}$

(f) Corresponding model for (e).

Figure 14.19: Example of chordal graphs generated by Procedure 14.1. $E(x, y)$ signifies the edges of the chordal graph as pairs, and the model M lists the elements of the domain.

14.7.2 Method #2: Recursive Model Generation

In this second approach to building models of MoSt, recall that all primitive functional groups are axiomatized in the ontology – we can leverage the fact that axiomatizations of molecules relies heavily on the fact that we *reify* functional groups in the ontology. First, we discuss a routine that takes a `molfile` and decomposes the underlying chemical graph into simple cycles and paths. Then we take the results of this decomposition procedure and map the cycles and paths with a library of functional groups. Finally, we take these *pieces* together and use a procedure to generate a model of the ontology. We describe each of these steps in turn.

Procedure 14.2: A Routine for Decomposition Using Breadth-first Search (BFS)[2]

To decompose molecules into their primitive functional groups, we re-emphasize that molecules can be considered as *topological graphs*. We are thus able to use a tailored breadth-first search algorithm to decompose the graph into its parts: rings and chains.

Breadth-first search is an algorithm that explores all possible vertices from a supplied set of vertices for a graph, as defined in Defintion 14.9. The algorithm returns all possible paths between the starting vertex and goal/ending vertex with the guarantee of returning the shortest path in the process. With this in mind, we leverage breadth-first search as

[2]The development of this decomposition routine into a set of Python scripts was Amit Sandhel's Master of Engineering (M.Eng) project at the University of Toronto. A copy of the Python scripts and project report can be found here: `http://stl.mie.utoronto.ca/amit_meng/`

the shortest path that is returned is what we need to determine simple cycles in the underlying graph (and thereby any primitive functional groups that are cyclic).

Breadth-first search is normally used to traverse or search trees and graph structures, with an arbitrary node as the starting point and the algorithm explores neighbouring nodes of the graph before moving on to the next level of nodes. The algorithm explores all possible vertices from the supplied set of vertices (in this case, the adjacency matrices shown in Chapter 5.1). With this algorithm, the shortest path is returned – this mirrors the fact that simple cycles are the shortest paths in the cyclic graphs.

> **Definition 14.9: Breadth-First Search (Adopted from [Cor+09; PM17])**
> *Assume graph $G = (V, E)$ is represented as adjacency lists. Attributes are attached to each vertex of the graph; the colour of each vertex $u \in V$ is stored in the attribute $u.colour$, and the predecessor of u is stored in the attribute $u.\pi$. If u has no predecessor, then $u.\pi = NIL$. The attribute $u.d$ holds the distance from the source s to vertex u computed by the algorithm. The algorithm uses a first-in, first-out queue Q to manage the set of grey vertices.*

We devised the following modifications to the breadth-first search algorithm, which is also summarized in Procedure 14.2:

- We assume the input of the algorithm is the molfile that contains the connectivity of the chemical graph in an adjacency list structure. A set of Python scripts were developed to parse the contents of a folder of `.mol` files and generate text output indicating the *simple* cycles found in each molfile.
- We assume that all simple cycles are the shortest paths for a given cyclic graph. Once the shortest path is found, the algorithm terminates.
- The algorithm utilizes *back-edge detection*, a characteristic of depth-first search. Once a path from the starting node to the end node is reached, adjacent vertices attached to the end node are analyzed to determine whether a back edge exists.
- When a path is identified as a cycle in the graph, the path is sorted and compared with the list of existing cycles found in the graph.
- If the cycle path already exists in the cycle list, then it is ignored, otherwise it is added to the cycle list and the algorithm continues until it terminates.
- Upon termination, all leftover paths that are not in cycles are also printed after the cycle list.

Procedure 14.2 is an abridged version of the Python scripts since there are multiple Python files and functions; the sub-procedures are named according to the Python functions developed by Amit. (For a more comprehensive discussion of the Python scripts, we refer the reader to the full project report.)

Procedure 14.2 molBFS

Input: `molfile`
Output: list of simple cycles and leftover paths in the `molfile` graph

```
1: procedure BFS_PATHS
2:     Generate graph from the adjacency table in molfile
3:     Store the indices and adjacencies
4:     Initialize a queue queue, a set start containing start node and enter list of
   nodes in the graph, a vertex vertex, and a set path
5:     Set queue = [(start, [start])]  ▷ initialize the queue with the start values
6:     while Q is not empty do                                  ▷ iterate over the queue
7:         (vertex, path) = queue.pop(0)            ▷ pop the first node from queue
8:         for nextNode in graph[vertex] - set(path) do    ▷ do a set operation of
   vertices in the graph against the set of paths
9:             if nextNode == goalNode then   ▷ if the node is in the path, append
   it to the list
10:                ans = path + [nextNode]
11:                if length(ans) > 2 then ▷ if length is greater than 2, it is not a chain
12:                    Call path + [nextNode]              ▷ call a generator function
13:                else                      ▷ if the vertex has length = 2, then continue
14:                end if
15:            else
16:                queue.append((nextNode, path + [nextNode]))
17:            end if
18:        end for
19:    end while
20:    return cycle and path list                                    ▷ in a .txt file
21: end procedure
```

Procedure 14.2 molBFS (continued)

```
22: procedure SHORTEST_PATH(START,GOAL)(returns the shortest path between start and goal)
23:     try: return next(bfs_paths(start, goal))
24:     if error then
25:         stop return none
26:     end if
27: end procedure
28: procedure CYCLE_RUNNER(runs BFS algorithm and produces a list/set)
29:     for item in edges do                                   ▷ Iterates over edges
30:         ans2 = shortest_path(item[0],item[1])
31:         if ans2 == none then     ▷ if there is no cycle in the graph, the list is none
32:             pass
33:         else
34:             ans2.sort()                                    ▷ Sort the list
35:             if ans2 in cycle_list then    ▷ if cycle already exists in the list, pass
36:                 pass
37:             else
38:                 cycle_list.append(ans2)
39:                 append the path to the list
40:             end if
41:         end if
42:     end for
43: end procedure
44: procedure MAIN(main Python class to call all of the scripts)
45:     cycle_finder(graph)                  ▷ calls cycle_finder to find all graphs
46:     cycle_list = []                  ▷ global variable that stores the cycles
47:     key_list = graph.keys()          ▷ global variable that stores graph info
48:     edges = list(itertools.combinations(key_list,2))      ▷ make combinations of edges
49:     cycle_runner()                                         ▷ calls cycle_runner()
50: end procedure
```

Procedure 14.3: Identifying Cycles and Paths as Primitive Functional Groups

Now that we have a method to determine the simple cycles and paths in a graph, we need to identify the corresponding rings and chains of these pieces of the graphs. Procedure 14.3 summarizes the steps taken to determine the corresponding primitive functional groups of these components of the graphs. The text file generated from Procedure 14.4 is used as the input of this identification procedure and first parses the cycle list input and is matched against a library of primitive functional groups to determine if the cycle is already known. This library of primitive functional groups was generated using Procedure 14.4 – a listing of the most common primitive functional groups were generated

to form the basis of the knowledge base. After all the simple cycles have been identified as ringed primitive functional groups and stored in another list, the procedure moves onto to the list of leftover paths and attempts to identify the paths as chained functional groups. The procedure terminates when the input file has been completely parsed.

Procedure 14.3 IdentifyGroups

Input: list of simple cycles and paths $L_1 = \{l_1, \ldots, l_n\}$, library of primitive functional groups `lib`

Output: list L_2 of corresponding primitive functional groups

```
1: procedure IdentifyGroups
2:     for each l_i in L_1 do
3:         if l_i not in functional group library then
4:             find corresponding functional group of l_i and add to L_2 ▷ add the cycle or ring to the library
5:             add l_i to L_2                    ▷ add the corresponding group to axiom
6:         else
7:             add l_i to L_2                    ▷ add the corresponding group to axiom
8:         end if
9:     end for
10:    return L_2                ▷ Return the list of corresponding functional groups
11: end procedure
```

Procedure 14.4: Recursively Building Models

A key advantage of axiomatizing MoSt in first-order logic is that we can utilize existing tools like Prover9 and Mace4 to do verification *and* build models. Since we have procedures to map simple cycles and paths to primitive functional groups, we can also build models of these primitive functional groups and skeletons using the ontology. Procedure 14.4 below outlines a recursive approach to building models of the functional groups and skeletons – the procedure continuously generates Mace4 models and combines these models together using an attachment procedure specified in Procedure 14.5.

With Procedure 14.4, we generate models of all *primitive* functional groups using Mace4. With a library of Mace4 models of primitive functional groups, we can then pick any two groups and attach them using the operators defined in Chapter 14.5.2 to get a combined model of the ontology. Recall that we utilize a declarative approach to leverage the spanning tree approach presented in Chapter 9.2, where every graph contains a spanning tree that allows us to characterize the tree and set of cycles with the ontology (and consequently we can use Procedure 14.2 to find the these simple cycles).

Procedure 14.5: Attaching Models Together

Furthermore, Procedure 14.5, is a separate procedure that combines models together by way of attachment graphs. The procedure takes in a chordal graph (generated from Procedure 14.1) and picks two vertices connected by an edge in the graph. Each selected

Procedure 14.4 RecursiveModelGeneration

Input: Primitive Functional Groups $G = \{G_1, G_2, ..., G_n\}$ and models $\mathcal{M} = \{Mod(G_1), Mod(G_2), ..., Mod(G_n)\}$

Output: Model $\mathcal{M} \in Mod(T_{most})$

1: **procedure** RecursiveModelGeneration
2: **for each** $G_i, G_j \in G$ **do**
3: **if** $Mod(G_i), Mod(G_j) \notin Mod(T_{most})$ **then**
4: generate $Mod(G_i)$ and $Mod(G_j)$ with Mace4 ▷ Generate models using a model builder
5: **else**
6: $\mathcal{M} = \mathfrak{attach}(Mod(G_i), Mod(G_j))$ ▷ Attach the models using $\mathfrak{attach}$
7: **end if**
8: **end for**
9: **return** $\mathcal{M}$ ▷ Return the combined model
10: **end procedure**

vertex is then substituted with a model of MoSt, and an exclusive-OR (XOR) is used to select an attachment operator (spiro, tether, or fusion) that will be substituted for the selected edge. The procedure ends when all vertices and edges have been selected and assigned models and attachment operators, and returns the combined model.

Procedure 14.5 attach

Input: Chordal Graph G generated from Procedure 14.1 or graph from Procedure 14.2

Output: Model $\mathcal{M} \in Mod(T_{most\_graph})$

1: **procedure** attach
2: Pick a vertex v_1 in G.
3: Pick an edge e_1 connected to the vertex v_1.
4: Pick another vertex v_2 connected to e_1.
5: Generate model $\mathcal{M}_i \in Mod(T_{most\_graph})$.
6: Select an attachment operator ($\oplus$ = XOR):

$$\mathcal{M} = \mathfrak{spiro}(\mathcal{M}_1, \mathcal{M}_2) \oplus \mathfrak{tether}(\mathcal{M}_1, \mathcal{M}_2) \oplus \mathfrak{fusion}(\mathcal{M}_1, \mathcal{M}_2)$$

7: **return** $\mathcal{M}$ ▷ Return the combined model
8: **end procedure**

14.7.3 Putting Everything Together

Using all of the procedures presented, we can summarize them in the flowchart in Figure 14.20. This flowchart mirrors the decomposition-composition framework presented in Figure 14.1, where we can consider the following:

- The *decomposition* of molecules into models corresponds to first utilizing Procedure 14.2 to identify the simple cycles and paths in the underlying graph of a molecule (in the form of a `molfile`). With these simple cycles and paths, we can then identify the corresponding primitive functional groups with Procedure 14.3. Then, we can generate the corresponding models of these primitive functional groups and leftover chains using Procedure 14.4. Finally, in Procedure 14.5, we can combine the functional group models together using the underlying chemical graph (from the `molfile`) as our blueprint for the molecule.

- The *composition* of models corresponds to the generation of an arbitrary chordal graph with Procedure 14.1. Using this chordal graph, we can generate a model by using Procedure 14.5 to get a model of MoSt.

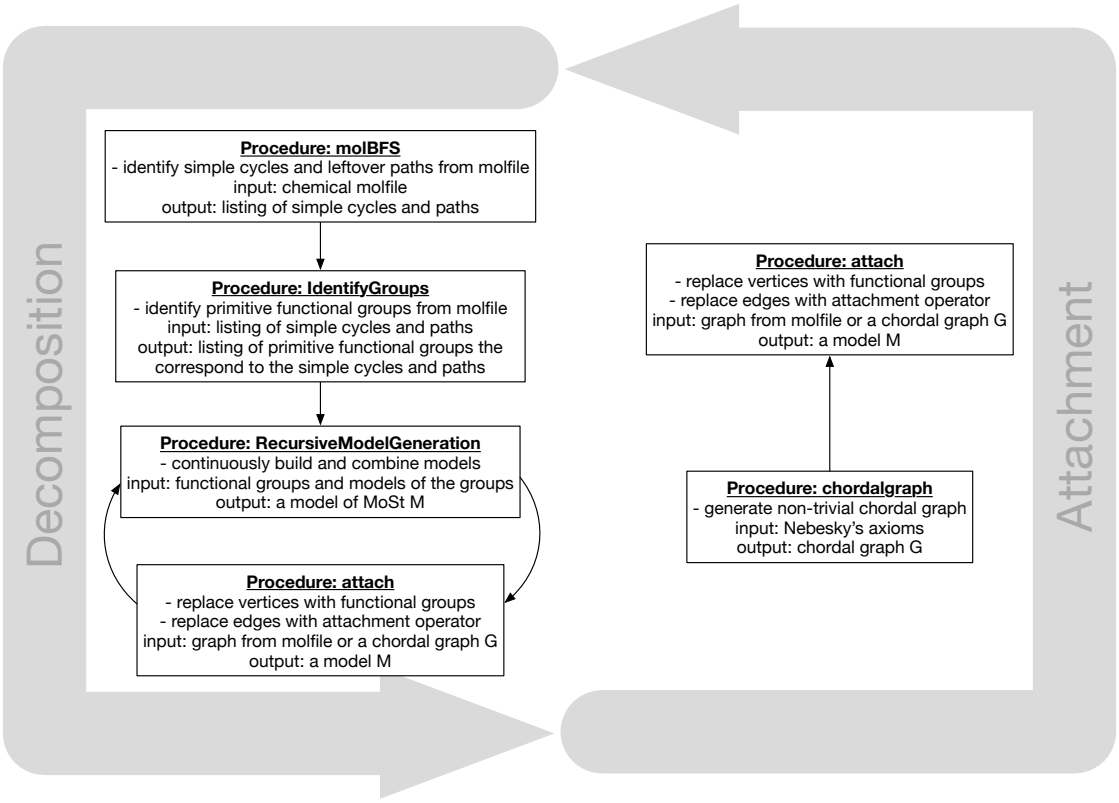


Figure 14.20: A flowchart for generating models via attachment. We can see here that the use of the procedures is similar to the framework presented in Figure 14.1.

14.8 Validation, Chemical Feasibility, Correctness & Completeness

We note here that we do not guarantee chemical feasibility with the axiomatization of molecules: this is due to the fact that the MoSt ontology deals with molecules from the perspective of the topological chemical graphs. One way of validating the models generated from the ontology is to do a consistency check with the generated models with already physically existing molecules.

Now that we have gone over the algorithm used to decompose an underlying graph into its parts, we are faced with the problem of how to handle the feasibility of graphs that are described by the ontology. We are faced with a validation problem that is composed of two parts:

1. Every molecule is a *model* of the ontology.
2. Every model of the ontology is a *molecule*.

Molecules = Models of MoSt For molecules that currently exist and are chemically feasible, these are named in accordance with the IUPAC naming rules. Recall that every organic compound has a name from which an unambiguous structural formula can be created [FDF74; Pur+93] – in this scenario, this means that every IUPAC has an associated `molfile` that can be used to extract the adjacency matrix (as seen in Figure 5.5(c)) which can be used to build the underlying chemical graph. From what we have seen in Part I, we are able to axiomatize the chemical graphs by taking the graphs and decomposing them into their primitive functional groups, and then re-composing them again via the attachment theorem presented in Chapter 14.5. With Theorem 14.5.2, we have shown that two models of the ontology can be combined together to form a model of the ontology – consequently, we can say with certainty that all IUPAC-named molecules that have underlying chemical graphs are therefore models of the ontology.

Models of MoSt = (Chemically Feasible) Molecules? Conversely, a model of MoSt is essentially a chemical graph: we can give this graph to a chemist and ask them to determine whether it is possible to synthesize the molecule. Whether or not a chemical graph can be physically realized is dependent on the difficulty of the synthesis process, along with the number of steps required.

We argue that a model of MoSt can result in *potential* molecules that have not yet been discovered and physically realized. We equate drug discovery to be the process of building models via answering queries with the ontology and constructing models of the ontology using additional constraints and extensions (we discuss this further in Chapter 15).

Correctness In order for a molecular description to be correct, it must accurately describe the molecule using the terminology of MoSt. To handle this, we utilize the existing representations of molecules that are stored as adjacency tables in the `molfiles`

that are readily available on public molecule databases such as ChemSpider[3] and PubChem[4]. Leveraging these molfiles, we are able to generate molecule descriptions by first decomposing the `molfile` into its primitive functional groups. Once decomposed, the resulting pieces of the graph can be then parsed and the molecule can be written as first-order axiom.

Completeness To ensure completeness of the molecule descriptions, we initially have generated the molecule descriptions and axioms for what we call the primitive functional groups. By placing restrictions on how these groups can combine and connect with each other, we are able to ensure that the combination of functional groups together is valid in the ontology.

K_4-free Graphs and Platonic Molecules With respect to feasible molecules, we note here that the models of the ontology can be considered K_4-free if enforced with an extension of the ontology that only allows graphs that are K_3 or smaller: this means that we can exclude graphs that are K_4 or higher (such as K_5, K_6, etc.) with such an extension of the ontology:

$$\begin{aligned}\neg(\exists a_1 \exists a_2 \exists a_3 \exists a_4 \exists b_1 \exists b_2 \exists b_3 \exists b_4 \exists b_5 \exists b_6\ & atom(a_1) \wedge atom(a_2) \wedge atom(a_3) \wedge \\ & atom(a_4) \wedge (a_1 \neq a_2) \wedge (a_1 \neq a_3) \wedge (a_1 \neq a_4) \wedge \\ & (a_2 \neq a_3) \wedge (a_2 \neq a_4) \wedge (a_3 \neq a_4) \wedge bond(b_1) \wedge \\ & bond(b_2) \wedge bond(b_3) \wedge bond(b_4) \wedge bond(b_5) \wedge \\ & bond(b_6) \wedge mol(a_1, b_1) \wedge mol(a_2, b_1) \wedge mol(a_2, b_2) \wedge \\ & mol(a_3, b_2) \wedge mol(a_3, b_3) \wedge mol(a_1, b_3) \wedge mol(a_4, b_4) \wedge \\ & mol(a_1, b_4) \wedge mol(a_4, b_5) \wedge mol(a_2, b_5) \wedge mol(a_4, b_6) \wedge \\ & mol(a_3, b_6))\end{aligned} \tag{K4-free}$$

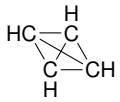
If we include such a constraint in the ontology, this means we prevent the tetrahedrane (C_4H_4) molecule, along with its derivatives, from being a model of MoSt (see Figure 14.21). Since this is a highly unstable molecule that is difficult to synthesize [Gün88; MOF14], we presume it is acceptable to leave it out of the permissible models of the ontology.

Existing research in [Gün88], [MOF14], and [Yuz+13] indicates that chemists are interested in synthesizing highly platonic[5] and unusual hydrocarbons (see Figure 14.22). However, due to bond strain, as well as the difficulty of synthesizing such molecules [MOF14], we make the assumption here in the ontology to *exclude* such hydrocarbons.

[3] `http://www.chemspider.com/`

[4] `https://pubchem.ncbi.nlm.nih.gov/`

[5] In three-dimensional space, a platonic solid is a convex polyhedron. Recall our earlier discussion in Chapter 2.1.2 on polyhedra.



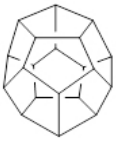
(a) Tetrahedrane as a K_4 graph

(b) 3D version of the molecule which resembles a tetrahedron discussed earlier Chapter 2.1.2

Figure 14.21: The tetrahedrane molecule (C_4H_4), depicted as a planar graph and drawn in three dimensions.

(a) Tetrahedrane (C_4H_{14})

(b) Cubane (C_8H_8)



(c) Dodecahedrane ($C_{20}H_{20}$)

(d) Triprismane (C_6H_6)

(e) Pentaprismane ($C_{10}H_{10}$)

(f) Octahedrane ($C_{12}H_{12}$)

Figure 14.22: Examples of platonic hydrocarbons.

14.9 Summary

In this chapter, we have framed the process of drug design as a model construction task and presented a framework for the decomposition of molecules into models of MoSt, as well as the composition of models of MoSt as molecules. We revisited the competency questions presented at the beginning of this work, and have presented how models of MoSt correspond to molecules, thereby satisfying Requirements (**MOD-1**), (**MOD-2**), (**MOD-3**), and (**MOD-4**). Further, we have presented graph-theoretic operations for molecular graphs and have mapped these operations with the attachment operators for models of the ontology. We also presented two methods of generating models with the ontology by way of attachment graphs and through a recursive approach of generating models of the ontology using the underlying chemical graph.

Chapter 15

Navigating Chemical Space

Now that we have all of the model construction theorems, we also wish to address how MoSt can assist with navigating chemical space with models. Part of our interest in developing a molecular structure ontology was to examine how it could be used in practice. In this chapter, we discuss the challenges with searching chemical space, and our attempts at overcoming the huge search space with the use of molecular constraints and molecular descriptions.

15.1 Chemical Space

In the most general sense, chemical space is the property space spanned by all possible molecules and compounds, where its size is unknown. In cheminformatics, chemical space refers to that of potentially pharmacologically active molecules, which are small enough to help modulate biochemical processes in a disease-modifying manner [RA12; BMG96]. The specificity of binding sites, along with the drug molecule's shape, polarity, and chemical functionality, allows for the success of small molecule drugs: small molecule drugs have enough structural and functional diversity for a variety of binding sites, along with the fact that they can be optimized for efficacy and safety in vivo [RA12; BW10; Hor+11].

While the theoretically-possible chemical space is large in number, the known chemical spaces have been continually updated and most open access collections are accessible to the public (see Table 15.1). These resources are invaluable to medicinal chemists, however, the number of *known* molecules does not provide any useful information on what potential molecules can be. The notion of a *chemical space* suggests a form of geographical map that illustrates the distribution of known and unknown molecules and their properties. In existing cheminformatics work, maps are created by assigning dimensions to a series of molecular descriptors, such as the following (list taken from [TC09]):

- 0D: bond counts, molecular weight, atom counts
- 1D - fragment counts, H-Bond acc/don, Crippen, PSA, SMARTS
- 2D - topological descriptors (Balaban, Randic, Wiener, BCUT, kappa, chi)

- 3D - geometrical descriptors (3D WHIM, 3D autocorrelation, 3D-Morse) and surface properties + COMFA
- 4D - 3D coordinates and conformations (JCHEM conformer, CORINA, gold set, Crystaleye)

While various different molecular descriptors exist, and any combination can be made to produce a chemical space map, this is tedious for a chemist to traverse. Figure 15.1 outlines some examples of spaces generated from the PubChem database using molecular quantum numbers (MQNs) presented by the authors of [RA12]. Other maps can be generated for various dimensions – we refer the reader to [RA12] for additional examples.

Table 15.1: Open-Access Databases for Small Molecules

Database	Information Stored	URL
PubChem	Chemical molecules and their activities against biological assays	https://pubchem.ncbi.nlm.nih.gov/
Chemspider	Chemical structure database provided by the Royal Society of Chemistry	http://www.chemspider.com/
ZINC	Commercially available chemical compounds for virtual screening	http://zinc.docking.org/
NCI Open	Cancer-related drug database	https://cactus.nci.nih.gov/ncidb2.2/
ChemDB	Commercially available small molecules	http://cdb.ics.uci.edu/
BindingDB	Bioactive molecules with binding affinity data	https://www.bindingdb.org/bind/index.jsp
CTD	Comparative toxicogenomics database	http://ctdbase.org/
HMDB	Human metabolome database	http://www.hmdb.ca/metabolites
SMPDB	Small molecule pathway database	http://smpdb.ca/view
DrugBank	Approved and experimental small molecule drugs	https://www.drugbank.ca/

Further, the work done in [Ehr13] shows that chemical patterns are generally used to search for substructures of molecules. These substructures tend to be specified as structural pattern languages like SMiles ARbitrary Target Specification (SMARTS) [Day07], the Molecular Query Language (MQL) [Pro+07], and the Sybyl Line Notation (SLN)

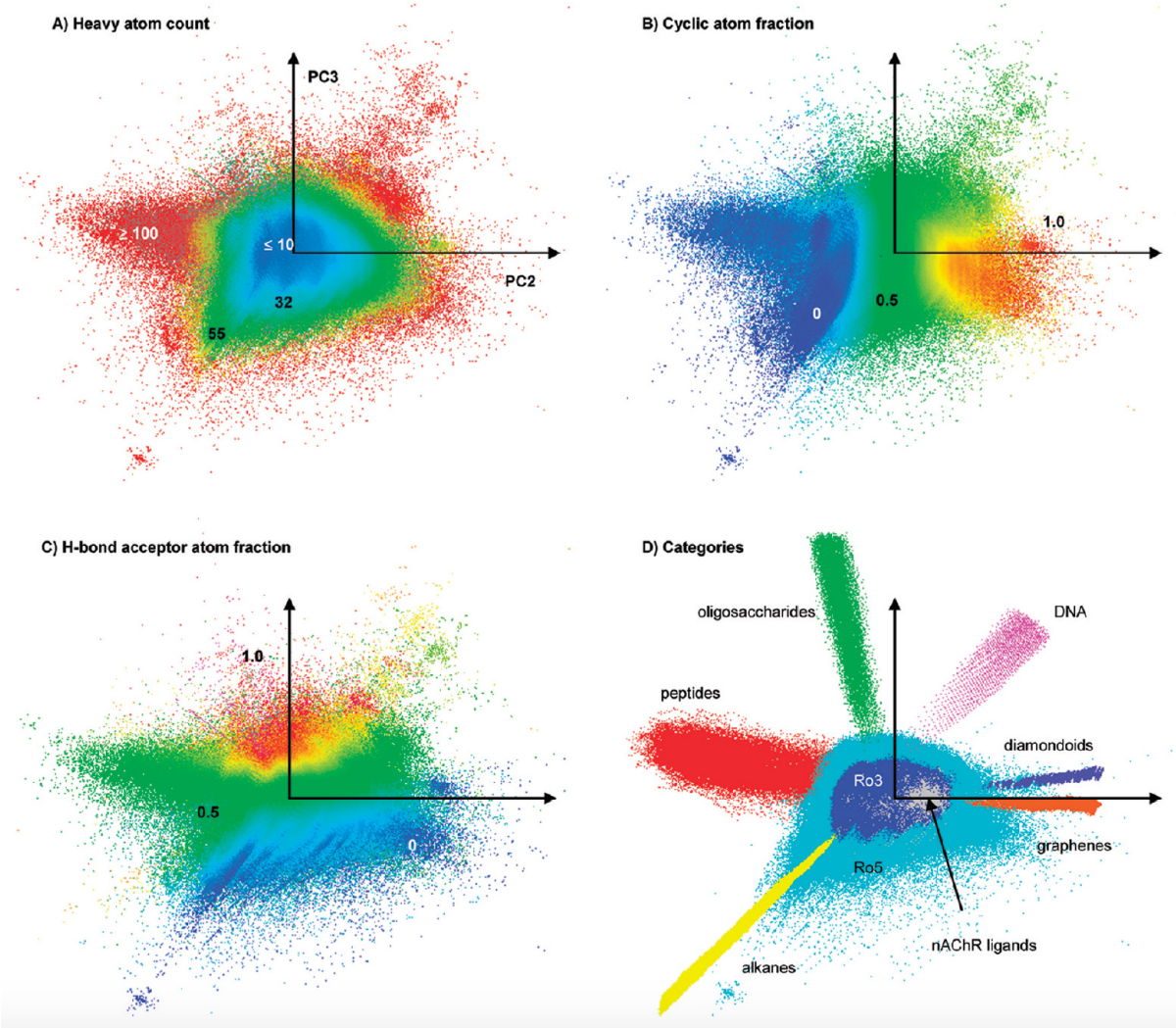
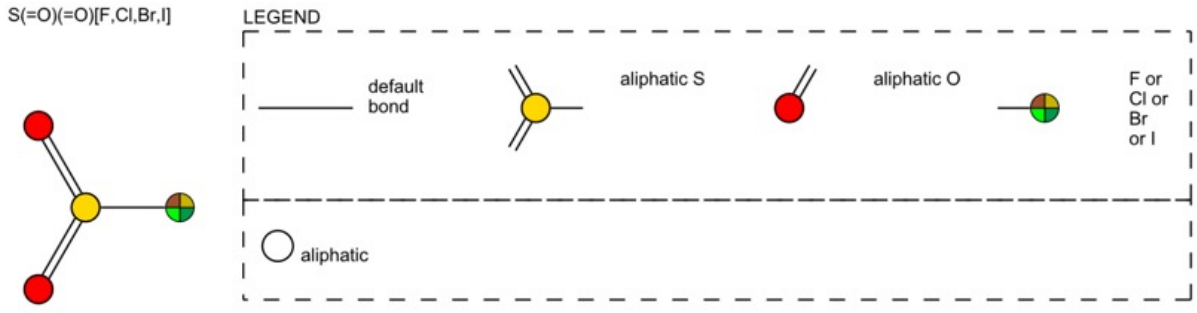
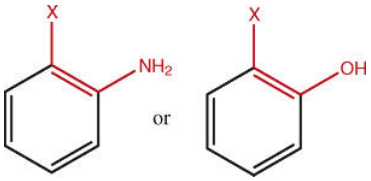


Figure 15.1: Colour-coded MQN-maps of the PubChem chemical space (Figure 1 in [RA12]). Each map contains a (PC2,PC3)-plane, which refer to the second and third principle component in the principal component analysis (PCA) of the MQN data for PubChem. (A) illustrates the map for the average number of non-hydrogen atoms per molecule, (B) illustratres the average fraction of cyclic atoms per molecule, (C) illustrates the average fraction of H-bond acceptor atoms per molecule, and (D) illustrates the compound categories including computationally enumerated molecules for each category.

[Hom+08]. SMARTS is an expressive line notation that utilizes SMILES to encode structures and was developed by Daylight Chemical Information Systems[1]. Similarly, MQL was developed to be a query language grounded on a context-free grammar, and unlike SMARTS, provides the specification of spatial properties of atoms and bonds. Finally, SLN is often compared with SMILES, as it is a specification language that describes molecules using short ASCII strings. Figure 15.2 outlines examples of how molecules and queries are described in the SMARTS, MQL, and SLN notations.

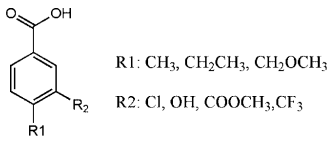


(a) Example of a SMARTS representation for sulfonyl halides generated using the SMARTSview tool.



Query: `Halogen-c:c-*[O&allHydrogens)1jN&allHydrogens)2].`

(b) An MQL query for a halogen substructure with an amine or a hydroxyl group. (Figure 5 in [Pro+07])



cSLN: C[1](C(C=O)OH):CH:C(G1):C(G2):CH:CH:@1\
{G1:CH3<v=1>|CH2CH3<v=1>|CH2OCH3<v=1>}\
{G2:Cl,OH,CH3OC(=O)<v=6>,FC(F)F<v=2>}

(c) An example of a combinatorial SLN for a Markush structure. (Figure 4 in [Hom+08])

Figure 15.2: Examples of substructure queries using SMARTS, MQL, and SLN.

Recall Figure 1.1 where one of the steps to producing a clinical candidate in the discovery pathways for drug discovery is the screening process (labelled as "Hit Finding: High-Throughput Screening (HTS) and High Content Screening (HCS)). During this stage of the drug discovery process, screening helps identify structures for a target protein. Compound libraries such as those listed in Table 15.1 are queried using various query languages. Screening is usually fully automated to test thousands of compounds in a short amount of time; the advancements of computing have allowed for HTS to be become an established procedure in drug discovery – the problem therein lies in the large amount of collected data that needs to be processed, along with navigating a chemical space size of 10^{60} molecules [BMG96].

In situations where it might be too costly to perform virtual screening, a pharmaceutical company may produce billions of compounds via virtual combinatorial chemistry. An efficient way of doing this is with the generation of *fragment spaces*: these are huge numbers of virtual compounds organized by their fragments and reagents, along with

[1] `http://www.daylight.com/dayhtml_tutorials/`

rules on how to combine them [Les+09]. Fragment spaces were first introduced as the Retrosynthetic Combinatorial Analysis Procedure (RECAP) in [Lew+98]; combinatorial rules define how to produce fragments from molecules, and how to combine them together. The work done in [MS07] outlines the key area that fragment lists and connection rules for fragments requires constant refinement over time due to the evolving nature of drugs and their uses.

From this discussion, we can see that the current approaches in cheminformatics and computational chemistry primarily focus on searching spaces utilizing strings of text using SMILES or other approaches to encode molecular structure in textual form. But have they considered how an ontology, along with a model-theoretic approach, might aid in this work? With respect to MoSt, by developing models of the ontology for the primitive functional groups, we can use these models to determine the models of molecules that consist of these primitive functional groups.

15.2 Using Queries & Models to Narrow the Search

With these chemical space maps in mind, we claim that navigating chemical space is analogous to solving *constraint satisfaction problems*: given specific conditions, a search for an optimal solution (in this case, a molecule) is similar to navigating the search space to satisfy a number of presented constraints.

15.2.1 Queries as Constraint Satisfaction Problems

Formally, a constraint satisfaction problem (CSP) is by a set of variables $\mathcal{V}$, a domain of values for each variable $dom(v)$ where $v \in \mathcal{V}$, and a set of constraints between variables $\mathcal{C}$. A solution to a constraint satisfaction problem (CSP) is an assignment of values to all variables that satisfies all of its constraints. Examples of CSPs include the eight queens puzzle (no two queens on an 8x8 chessboard threaten each other), map colouring problems (no two adjacent vertices are of the same colour), and Sudoku. Instead of looking at chemical space in terms of these SMILES encodings, we can consider the chemical search space as a knowledge base of all possible models that can be queried or created. Variables in the problem correspond to the relations in the ontology, the domain of values for each variable become the instances of these relations or classes, and the sets of constraints can become the actual query. With this analogy in mind, models of MoSt that are being generated with the ontology are not *arbitrary* models that satisfy the MoSt axioms. Rather than constructing arbitrary models of MoSt, we can generate models wherein the molecular descriptions are *complete*.

We briefly discussed molecular descriptions in Chapter 14.3 – here we introduce the notion of *complete* molecular descriptions in Definition 15.1, where we consider an axiomatization of a molecule such that its only model that satisfies the description is the molecule itself. In other words, the only possible model of a given molecular description is the molecule.

Definition 15.1: (Complete) Molecular Description
A complete theory that axiomatizes a model of MoSt: an axiomatization of a

molecule such that the only model of the molecular description is that molecule.

$$\textit{molecular description of } \mathcal{M} \textit{ is } Th(\mathcal{M})$$

Definition 15.2: Molecular Constraint
Weaker than molecular description, a molecular constraint is a sentence written in first-order logic that allows for multiple models to satisfy the sentence. It is a sentence in the signature of MoSt.

$$\textit{a sentence } \in \Sigma(MoSt)$$

We discuss how models of the ontology come into play with navigating chemical space by constraining the search space to explore smaller sets of potential molecules, instead of utilizing an ad hoc approach to search chemical space.

Generating Molecular Descriptions Recall our earlier discussion of molecular descriptions in Chapter 14.3, where we gave an overview of how the descriptions are conservative definitions that utilize the terminology of the ontology. We can use generated molecular descriptions to validate the ontology with the real world.

The generation of molecular descriptions from `molfiles` allows us to further demonstrate how the ontology is a commonsense approach to encoding additional information about the molecules that we normally do not see in any of the cheminformatics approaches discussed in Chapter 2.

With the modified breadth-first search procedure in Procedure 14.2, additional code was written in the Java programming language to parse the outputs containing the primitive functional groups into molecular descriptions. Procedure 15.1 takes in the text file from Procedure 14.2 as input and compares the simple rings and chains with existing the library of existing primitive groups, and then constructs the axiom in the Common Logic syntax based on the adjacency matrix in the molfile.

Procedure 15.1 Molecular Description Generation

Input: `molfile`, `.txt` output from Procedure 14.2
Output: `axiom` in Common Logic

1: **procedure** MOLECULAR_DESCRIPTION_GENERATOR
Require: Initialize a `queue` with a list of starting points
2: **while** `txt` is not empty **do**
3: find corresponding group
4: concatenate axiom C in Common Logic
5: **end while**
6: **return** `axiom` $\mathcal{C}$ ▷ Return the molecular description
7: **end procedure**

Recall the decomposition and attachment theorems presented in Chapter 14.5, we can

also update Figure 14.20 to include this molecule description generation procedure. In Figure 15.3 below, we present the updated procedures where the outputs of the decomposition and attachment procedures are fed into Procedure 15.1 and outputs the molecule description in the Common Logic syntax.

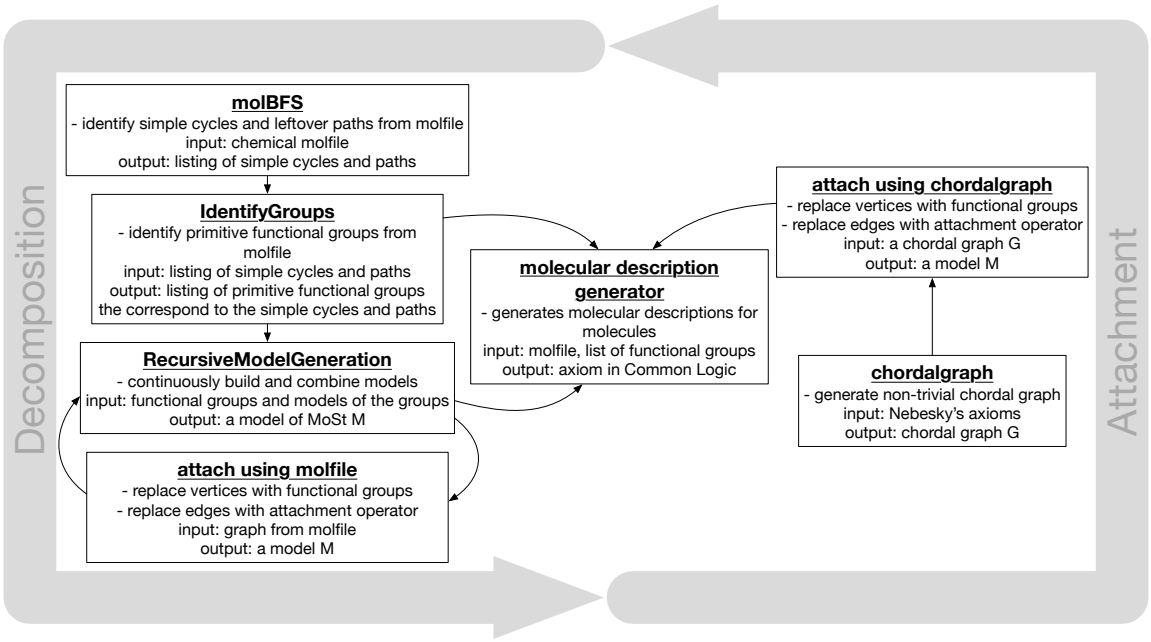


Figure 15.3: An updated version of Figure 14.20 for generating models via attachment with the inclusion of Procedure 15.1. The outputted models can be processed by Procedure 15.1 to generate molecular descriptions in Common Logic.

With model construction, we can treat axioms as part of the ontology or part of the query. With queries, we are interested in adding constraints to the ontology to build a model that satisfies the first-order sentence. Thus, we can look at drug design in these contexts:

1. **Information retrieval**: querying a knowledge base (KB) of existing molecules via **entailment**

$$KB \models (\text{molecular description or molecular constraint})$$

2. **Generating molecules**: solving satisfiability queries using MoSt by **building models** that satisfy molecular constraints

$$T_{most} \cup (\text{molecular constraint}) \text{ is consistent}$$

$$T_{most} \not\models \neg(\text{molecular constraint})$$

We discuss each of these below.

15.2.2 Retrieving Models via Queries (Information Retrieval)

One of the most common uses of an ontology is to answer questions via the form of queries. Using MoSt, we can query a knowledge base (KB) to retrieve models of a molecule by simply *searching* for the existing molecule. If the molecule currently exists, then it should already be axiomatized in the knowledge base. These queries are in the following format, where the molecular description or molecular constraint is entailed by the knowledge base:

$$KB \models \text{(molecular description or molecular constraint)}$$

For example, a query to retrieve models may be of this form, where the model we wish to retrieve would be for the daraprim molecule previously shown in Figure 14.3:

$$KB \models \exists x \, daraprim(x)$$

These queries are fairly simple in nature and do not provide any other insight other than retrieving models for given molecules. What about situations where the queries are incomplete? In other words, what if a medicinal chemist wants to find molecules with particular features but do not know exactly which molecules they want? In the context of MoSt, this would be a satisfiability problem with incomplete queries using molecular constraints.

15.2.3 Constructing Models with Molecular Constraints

As we've introduced molecular constraints in Definition 15.2, we consider queries that are *incomplete*, in the sense that we consider molecules that satisfy one or more molecular constraint. As mentioned previously, complete molecular descriptions are those that generate a molecular description for a *unique* molecule. With molecular constraints, we wish to construct models that satisfy these constraints.

Satisfiability queries are queries where models need to be created to satisfy the constraints or sentences. The format for these satisfiability problems are:

$$T \cup \text{(a constraint) is consistent}$$

In English, this can be read as "a theory T and the union with a constraint is consistent" – this means that a model generated with the union of T and the constraint demonstrates the consistency of this union. We can rewrite this query in the following format, which means that the theory does not entail the negation of the constraint:

$$T \not\models \neg\text{(a constraint)}$$

In the context of MoSt, these molecule construction queries are in the following format:

$$T_{most} \cup \text{(molecular constraint) is consistent}$$

$$T_{most} \not\models \neg\text{(molecular constraint)}$$

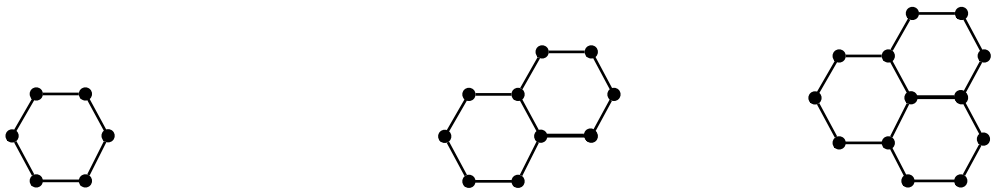
An example of a very simple satisfiability query would be to construct molecules that only have one ring:

$$T_{most} \not\models \neg(\exists x\, ring(x))$$

Another example would be to construct molecules that have fused rings:

$$T_{most} \not\models \neg(\exists x \exists y\, ring(x) \wedge ring(y) \wedge fused(x, y))$$

Figure 15.4 below provides some examples of models that satisfy these queries.



(a) A single ring. **(b)** Two fused rings. **(c)** Three fused rings.

Figure 15.4: Examples of models that satisfy incomplete queries pertaining to rings. Both (a), (b), and (c) satisfy $T_{most} \not\models \neg(\exists x\, ring(x))$, while (b) and (c) satisfy $T_{most} \not\models \neg(\exists x \exists y\, ring(x) \wedge ring(y) \wedge fused(x, y))$ since there are at least two rings that are fused together in both models.

From this, we can consider queries to be of the existential and universal format.

Existential Queries Consider queries where *only* existential quantifiers are used, such as:

- Construct a molecule that contains at least one ring.

$$T_{most} \not\models \neg(\exists x\, ring(x))$$

- Construct a molecule that contains at least one chain.

$$T_{most} \not\models \neg(\exists x\, chain(x))$$

- Construct a molecule that contains at least one fork.

$$T_{most} \not\models \neg(\exists x\, fork(x))$$

- Construct a molecule that contains at least one fork and at least one ring.

$$T_{most} \not\models \neg(\exists x \exists y\, (x \neq y) \wedge fork(x) \wedge ring(y))$$

- Construct a molecule that contains a ring that is fused to another ring.

$$T_{most} \not\models \neg(\exists x \exists y\, (x \neq y) \wedge ring(x) \wedge ring(y) \wedge fused(x, y))$$

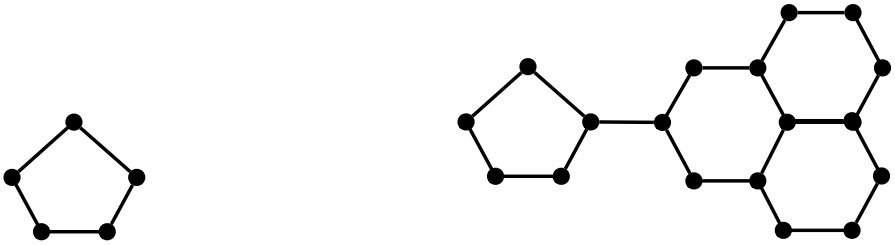
- ...and so forth...

As we can see, these existential queries are fairly simple, as the only models that satisfy these existence conditions need to be constructed. By satisfying an existential sentence, every extension of this sentence can become a model; for example, if we have a query that indicates there exists a ring in the molecule, then any additional constraints to the query will be satisfied regardless:

$$\text{if } T_{most} \not\models \neg(\exists x\, ring(x)) \text{ is satisfied, then}$$

$$T_{most} \not\models \neg(\exists x\, ring(x) \wedge \text{...additional constraints...}) \text{ will also be satisfied}$$

We illustrate this in Figure 15.5, where (a) is a simple model that satisfies the query where a ring exists; this query is also satisfied in (b) even though there are four rings in the model (three are fused together while one is tethered).



(a) A ring that satisfies the query. **(b)** A tethered ring that also satisfies the query.

Figure 15.5: Both (a) and (b) satisfy the queries: $T_{most} \not\models \neg(\exists x\, ring(x))$ and $T_{most} \not\models \neg(\exists x \exists y \exists b\, ring(x) \wedge tether(b, x, y))$.

These types of queries are also known as conjunctive queries in database theory, which is also a query type used in relational databases.

Universal Queries Furthermore, we can restrict the search space with queries that deal with the universal quantifier; for example:

- Construct molecules where all groups are rings.

$$T_{most} \not\models \neg(\forall x\, group(x) \supset ring(x))$$

- Construct molecules where all groups are chains.

$$T_{most} \not\models \neg(\forall x\, group(x) \supset chain(x))$$

- Construct molecules where all groups are fused.

$$T_{most} \not\models \neg(\forall x \forall y\, group(x) \wedge group(y) \wedge (x \neq y) \wedge fused(x, y))$$

- Construct molecules where all groups are tethered.

$$T_{most} \not\models \neg(\forall x \forall y\, group(x) \wedge group(y) \wedge (x \neq y) \wedge tethered(x, y))$$

Combination of Existential and Universal Queries Additionally, we can also include queries with mixed quantifiers, such as the following:

- Construct molecules where all groups are fused rings.

$$T_{most} \not\models \neg(\forall x\, group(x) \supset ring(x) \wedge \exists y\, ring(y) \wedge (x \neq y) \wedge fused(x, y))$$

- Construct molecules where all groups are tethered to at least one other group.

$$T_{most} \not\models \neg(\forall x\, group(x) \wedge \exists y\, group(y) \wedge tether(x, y) \wedge (x \neq y))$$

15.2.4 Using Extensions of the Ontology to Constrain Search Space

Consider our earlier presentation of the ontology in Chapter 8, we can use extensions of the ontology to limit the classes of molecules we want to examine. Recall the notion of a conservative extension from Definition 11.4. Suppose we wish to restrict terminology of the ontology through the use of extensions of the ontology: this allows us to capture and reuse existing constructions of MoSt models. If we know that one theory is a conservative extension of another through proof-theoretic means, then we can determine that the models of the conservative extension can be constructed from models of the other.

Suppose we only want to look at the class of polycyclic molecules and their properties. Recall that polycyclic molecules are molecules that contain one or more ring. We can introduce an extension of MoSt, $T_{most\_polycyclic}$[2], that contains only one axiom which constrains that all groups as rings in the ontology:

$$\forall x\, (group(x) \supset ring(x)). \tag{MP-1}$$

We can limit the knowledge base to only handle polycyclic molecules by combining this extension with the regular MoSt ontology via the union of the ontology and the extension, $T_{most} \cup T_{most\_polycyclic}$, as follows:

$$T_{most} \cup T_{most\_polycyclic} \not\models \neg(\text{your query here...})$$

For example, if we want to find all polycyclic molecules that are fused together, we do not need to add in the condition that all groups are rings (since it is already known that $T_{most\_polycyclic}$ is an extension of MoSt where all groups are rings):

$$T_{most} \cup T_{most\_polycyclic} \not\models \neg(\forall x \forall y\, group(x) \wedge group(y) \wedge (x \neq y) \wedge fused(x, y))$$

This further constrains the query to only search the class of polycyclic molecules. With respect to COLORE, if we have constructed the models of one theory T_2 (e.g., $T_{polycyclic}$) by extending models of the other theory T_1 (e.g., T_{most}), then this is captured in COLORE as a conservative extension relationship. Other examples of extensions to MoSt can be found in Appendix C.2.

[2] `http://colore.oor.net/most/most_polycylic`

15.3 Narrowing the Search with Models

Current practice with narrowing the chemical search space is somewhat directed with the usage of scaffolds , but we can leverage the ontology and query satisfaction to help restrict the search space. As we have seen in the previous section, we can adjust both sides of the entailment process:

1. We can add additional constraints on the consequent (right-hand side) of the query by introducing more constraints that need to be satisfied. These constraints were previously shown in Chapter 15.2.2.

$$T_{most} \not\models \neg(\text{molecular description or molecular constraint} \cup \text{additional constraint})$$

2. We can create extensions of the ontology on the antecedent (left-hand side) of the query to limit the knowledge base. These constraints were discussed in Chapter 15.2.4.

$$T_{most} \cup T_{most\_extension} \not\models \neg(\text{molecular description or molecular constraint})$$

In practice, what these queries mean is that:

- The more incomplete the query (molecular constraint), the higher number of possible models can be satisfied.
- The more complete the query (molecular description), the smaller number of possible models can be satisfied.

This idea of navigating chemical space is analogous to *model theory* concepts called universal formulae, universal theories, and existential formulae. We leveraged these notions in the previous section when discussing the existential, universal, and hybrid queries to narrow the search space for molecules. These model-theoretic notions are summarized as follows (adapted from [WD15]):

1. A *universal formula* is a formula $\phi(x)$ of the form:

$$\forall y\, \phi(x; y)$$

with $\phi(x; y)$ quantifier-free. A *universal theory* is a theory that consists of universal sentences.

2. An *existential formula* $\phi(x)$ is of the form:

$$\exists y\, \phi(x; y)$$

with $\phi(x; y)$ quantifier-free. Similarly, an *existential theory* is a theory that consists of existential sentences.

With these notions, this means that, for a universal theory T that has a model $\mathcal{M}$, any substructure of $\mathcal{M}$ is a model of T. This is particularly important with MoSt as molecular

constraints are generally formulated as logically *weaker* sentences to allow for higher number of models that satisfy the constraints. Conversely, a molecular description is a logically *stronger* axiomatization of a molecule such that the *only* model of the definition is the molecule itself.

15.4 Summary

In this chapter, we have discussed the current work done to search chemical space for small molecules that satisfy certain requirements (small, drug-like, exhibits certain shapes/properties, and so forth). We have also presented an analogy of searching chemical space by using MoSt to solve constraint-satisfaction problems (queries).

Chapter 16

Conclusions

From the previous chapters, we have identified the need for a molecular structure ontology. Through an examination of the existing conventional cheminformatics and ontological approaches, we have identified formal requirements for a molecular structure ontology that combines these approaches together. With a presentation of the ontology's axioms, and a discussion on the technical results of verifying the ontology with mathematical theories, we have identified graph theoretic notions that also provide a new approach to designing new molecules through model construction. Here we summarize the key contributions of this work, lessons learned, and discuss areas of future work.

16.1 Contributions

In this work, the following contributions have been made:

Design and Verification of MoSt (Chapters 4 to 13). Over the course of Parts I and II of this thesis, we have presented a first-order axiomatization of molecular structure, in the context of static, topological chemical graphs. We have shown how existing work in ontologies and cheminformatics fall short of truly axiomatizing molecular structure: existing cheminformatics ontologies focus more on classification and taxonomies of molecules, whereas existing ontologies focus on spatial properties of molecules [Coh01], rules for classification [MKH14], or utilize a higher order logic that do not have reasoners [HKM11].

In Part I, we provided an overview of the ontology, along with the key design decisions made: functional groups are elements of the domain, and we permit three attachments between groups: fusion, spiro, and tethering. It was essential that functional groups were elements of the domain, as existing work done in chemical graph theory only looks at a chemical graph in terms of its atoms and bonds, but not groups. We also made the decision that underlying chemical graphs are loopless, nonisolated multigraphs to ensure that unsaturated groups and compounds are properly represented from a graph-theoretic perspective. Additionally, we placed limitations on how to properly represent chemical bridges in a molecule: compounds are allowed to be multiply-fused, whereas the spiro and tether attachment relations are only permitted to be unique (connected at exactly

one atom or by one bond). Furthermore, we discussed how spanning trees and ringbonds ensure that *simple* cycles in a graph correspond to chemical rings. Finally, we presented the notion of skeletons that combined all of the concepts and attachments together.

To ensure that the ontology was consistent and contains models that correspond to our intended models, we verified MoSt in Part II. We first discussed how incidence structures in geometry make up the intended models of MoSt, and presented the graph-theoretic notions needed in these incidence structures. The axioms of MoSt were then grouped into interconnected theories, two of which were verified using Prover9: T_{most_graph} and T_{most_group}. The former summarizes existing chemical graph theory notions, and the latter summarizes the major components of MoSt: functional groups are elements of the domain. From the verification results, we have shown that both theories of MoSt are definably equivalent to two theories of incidence structures found in the COLORE repository, $T_{nonisolated\_loopless}$ and $T_{cycle\_path\_subgraph\_nonisolated}$. Finally, we presented representation theorems of the T_{most_graph} and T_{most_group} theories of MoSt.

New Techniques for Designing Molecules via Model Construction (Chapter 14). We presented a new technique for designing molecules via model construction: since molecules can be decomposed into and composed of primitive functional groups, it follows from having an ontology of molecular structure that it would be possible to reflect this with first-order models of the ontology. We have presented two structure theorems for the models of MoSt: a decomposition theorem that breaks down existing molecules into their primitive functional groups, and an attachment theorem that indicates how primitive functional groups can be combined together using attachment operators. Furthermore, we have shown how these attachment operators for the fusion, spiro, and tether attachments in graphs correspond to model-theoretic operators for the combination of models.

Using chordal graphs as the basis for model construction, we introduced the notion of attachment graphs that can be used to build arbitrary models of molecules when used in conjunction with a model builder like Mace4. Further, we introduced several procedures to decompose (existing) molecules into models of MoSt by utilizing a modified breadth-first search algorithm to identify simple cycles in an underlying chemical graph as ringed groups, and simple paths as chained groups. With this algorithm, we presented procedures that can be used with Mace4 to build models of MoSt.

An Alternative Approach to Navigating Chemical Space (Chapter 15). In addition to providing new techniques for model construction, we presented an alternative approach to navigating chemical space via querying models. Existing work in computational chemistry has focused primarily on developing querying techniques and languages to search structures based on their encodings (either with SMILES, SMARTS, MQL, or SLN). We presented an approach where, instead of using these query languages, a user queries a knowledge base using molecular descriptions and molecular constraints written in the language of MoSt. We noticed similarities of searching chemical space with solving constraint satisfaction problems and discussed how framing a query using universal or existential constraints help narrow the search space, and provided examples of such

queries that can be used.

16.2 Lessons Learned

Being the designer of an ontology gives one many freedoms – as the designer, we are free to choose how to axiomatize concepts, but part of the responsibility is to present the ontology's axioms clearly to all intended users. In the case with MoSt, it was a challenge to present this to both the chemistry and the ontology communities: with writing this dissertation, we needed to be cognizant of the fact that chemists probably have not come across ontologies in their work; similarly, since MoSt is a domain-specific ontology, not all ontologists know the subtle nuances in molecule naming in chemistry, or some of the model- and graph-theoretic concepts. Having presented initial iterations of MoSt to both communities, we became more aware of the importance of the overall narrative of the project: framing the uses and benefits of the ontology becomes important, regardless of the audience. Furthermore, as seen in Part III, MoSt combines concepts from multiple disciplines: chemistry, knowledge representation and reasoning, and graph theory. It becomes especially important in this case that we convey all of the multidisciplinary influences clearly and concisely to readers of this work.

16.3 Future Work

While MoSt axiomatizes small molecules, it is only a first step to creating ontologies for medicinal chemistry. Moving forward, areas of future work include extending MoSt with a molecular reactions ontology, developing a software environment for reasoning about molecular shape and reactions, and potential integrations with other cheminformatics software tools.

Open Questions with MoSt: Decidability, Mereology, & Planar Graphs Since our interest lies in the representation of structure, we have not determined whether MoSt is decidable – this might be in part due to the fact that we have chosen to axiomatize the ontology in first-order logic, which is also undecidable [Ben12]. With respect to the mereology on skeletons, we are currently unsure *which* classical mereological sum should be used to handle the combination of skeletons in MoSt – at the moment, this remains an open question that will need to be explored in more detail since it involves the verification of $T_{most\_skeleton}$ and an examination of how molecular structure changes with reactions. Furthermore, our earlier discussion about excluding planar molecules like tetrahedrane in the ontology poses an area of future work for the ontology: we would also like to axiomatize molecules whose graphs are planar. Since MoSt axiomatizes graphs that are K_4-free, we would need to examine how we can extend the ontology or add additional axioms to represent these underlying planar graphs.

Using MoSt to Axiomatize the Scaffold Tree Approach In chemistry, there is a concept of a 'chemical scaffold' that is the common *core* structure that characterizes

a group of molecules in which it is contained as a substructure [Sch+07]. Structures that share a scaffold are thought to have a common synthetic pathway [STW12], so compounds in combinatorial libraries are generally grouped and based on common scaffolds. Scaffolds are used to define classes of chemical compounds found in patent claims and are also referred to as *Markush structures*. In [Sch+07], the authors present a classification procedure that allows chemists to determine the scaffolds of compounds. An example of the pruning of molecules is presented in Figure 16.1, with a condensed list of the prioritization rules implemented in the Molinspiration toolkit[1]; for further reading on the pruning steps, please see [Sch+07]. The classification system consists of thirteen prioritization rules that need to be followed, and follows the notions of molecular frameworks discussed in [BM96]. To classify scaffolds, scaffolds are dissected using an iterative removal of rings, until a single "root" is obtained from the procedure. This permits chemists to have a hierarchical classification of scaffolds, where each scaffold is a well-defined chemical entity. A disadvantage of classifying scaffolds in this manner is that a classification tree of scaffolds can only have *one* parent scaffold.

Scaffold hopping (also known as *lead hopping*) is a strategy employed by medicinal chemists to discover structurally novel compounds [STW12]. A known active compound is taken and modified structurally to retain a novel chemotype or property in this process of scaffold hopping [STW12; Sch+07]. For chemists, it is of interest to group compounds based on common core scaffolds and substitution patterns: to do this, they must be able to categorize sets of scaffolds. We can use MoSt as the basis for an ontology for scaffold trees – essentially what we are interested here is to leverage the existing MoSt ontology and use it to help us define scaffolds and their associated terminologies. We can write definitions using the ontology's terminology for scaffold concepts. For example, a definition for a scaffold using MoSt terminology is shown in Definition 16.1 below.

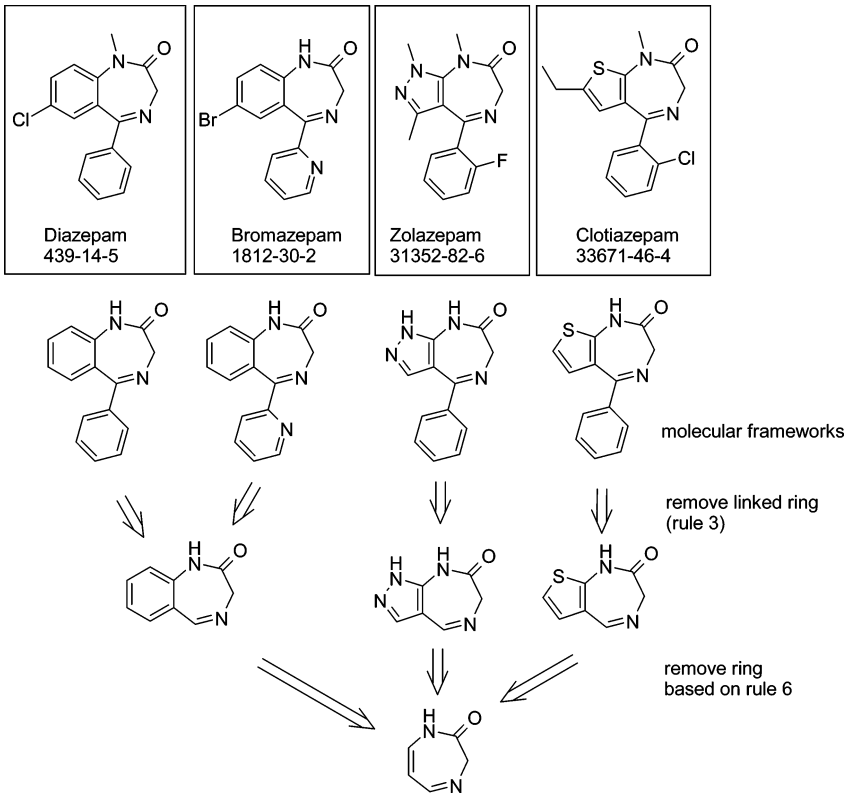
Definition 16.1: Scaffold (Adopted from [Pat13])
A scaffold is the core structure of a molecule round which variations are possible through the use of different substituents.

Using the terminology of MoSt, we can define scaffolds as skeletons that consists of at least one ring, and chains that are connected to rings in Axiom MSCD-1.

$$\forall x\,(scaffold(x) \equiv (skeleton(x) \wedge \exists r((ring(r) \wedge mol(r,x)))\wedge \\ \forall g(((chain(g) \wedge mol(g,x)) \supset\ \exists y \exists z((ring(y) \wedge ring(z)\wedge \\ mol(y,x) \wedge mol(z,x) \wedge attached(y,g) \wedge attached(z,g))))))). \quad \text{(MSCD-1)}$$

With this in mind, an area future work would be to utilize MoSt to create a *definitional extension* of the ontology with axioms describing the scaffold tree approach and all of the scaffold-related terminology.

[1] http://www.molinspiration.com



(a) Pruning with diazepam $C_{16}H_{13}ClN_{2O}$, bromazepam $C_{14}H_{10}BrN_3O$, zolazepam $C_{15}H_{15}FN_4O$, and clotiazepam $C_{16}H_{15}ClN_2OS$. After pruning, the seven membered diazepinenone ring becomes the scaffold for these diazepinenones. (Scheme 18 from [Sch+07].)

1. Remove heterocycles of size 3 first.
2. Do not remove rings with greater than 12 atoms if there are still smaller rings to remove.
3. Choose the parent scaffold having the smallest number of acyclic linker bonds.
4. Retain bridged rings, spiro rings, and non-linear ring fusion patterns with preference.
5. Bridged ring systems are retained with preference over spiro ring systems.
6. Remove rings of sizes 3, 5, and 6 first.
7. A fully aromatic ring system must not be dissected in a way that the resulting system is not aromatic any more.
8. Remove rings with the least number of heteroatoms first.
9. If the number of heteroatoms is equal, the priority of heteroatoms to retain is $N > O > S$.
10. Smaller rings are removed first.
11. For mixed aromatic/non-aromatic ring systems, retain non-aromatic rings with priority.
12. Remove rings first where the linker is attached to a ring heteroatom at either end of the linker.
13. Tiebreaking rule: remaining ties are solved by removing possible remaining sub-scaffolds, using canonical SMILES, that have a lower rank in alphabetical order.

(b) Scaffold generation rules from [Sch+07]

.

Figure 16.1: Example of pruning a molecule to determine its scaffolds.

Molecular Reactions Ontology (MoRe): A Process Ontology An area of further work would be to look into how MoSt can be extended to describe chemical reactions between compounds. As seen in medicinal chemistry, reactions play a big role in drug development since molecules undergo structural changes during reactions. By having a molecular reactions ontology, it would not only complement the MoSt ontology but would provide users of the ontology with further insight on the combinations of compounds for drug discovery. For example, developing MoRe can help enhance and promote the usage of medical ontologies, such as the Biological Pathway Exchange Ontology (BioPAX)[2], to describe chemical reactions of drugs in biological pathways.

To develop a process ontology for chemical reactions, we can utilize existing techniques from [Aam16], where the author discusses the action theories for domain-specific process ontologies. Since chemical reactions are numerous, it would be an interesting ontological challenge to specify a complete classification of possible activities within the domain of chemical reactions. Action theories are utilized in manufacturing, along with the Process Specification Language (PSL), to represent the process of cutting and welding 2D sheets [GB11]. A similar process of combining shapes together can be found in the chemical reactions of heterocyclic compounds, where sheet metal cutting processes may correspond with elimination reactions, in which atoms are removed from a molecule in a one- or two-step organic reaction. Similarly, welding sheets together may correspond with coupling reactions in organic chemistry, where two hydrocarbons are joined together with a catalyst. Due to these similarities with manufacturing, we would be interested in developing a molecular reactions ontology to represent these types of processes, and identify any relationships with work done in existing process ontologies.

Reasoning About Molecular Entities (RoME): A Software Environment Similarly, a complement to the MoSt ontology would be to have a software environment established for reasoning about molecular compounds and their reactions. It would be of interest to have some sort of software environment where a medicinal chemist can take the MoSt and MoRe ontologies and make semi-automatic deductions of structural properties and reactions for molecular compounds. A proposal would be a software environment called Reasoning About Molecular Entities (RoME). Integration with existing chemical software systems would be of interest as existing cheminformatics software rely heavily on the data formats that store structural information – integrating software applications with MoSt would be a semantic interoperability task that would be greatly aided with *a priori* knowledge in the form of ontologies.

Additionally, a reasoning environment such as RoMe would supplement existing cheminformatics techniques. This is especially important with predicting side effects of drug-like compounds; in the drug discovery timeline presented in Figure 1.1, we can see that testing a drug requires a multitude of *in vitro* and *in vivo* studies before it can be accepted by governing bodies. Having a reasoning environment that models potential reactions may also help with making predictions of drug effectiveness and (un)intended side effects. Being able to make predictions of drug efficacy might also be another application area or extension of MoSt with Molecular Reactions Ontology (MoRe).

[2] `http://www.biopax.org/`

Integration with (Cheminformatics) Software Tools & Query Languages With the rising interest in machine learning techniques, the development of MoSt can be seen as a stepping stone for *a priori* knowledge to help with machine learning: an ontology can serve to supplement and guide the learning process. One can also think of MoSt as part of a medicinal chemical assistant, where the ontology aids in explaining *why* a particular drug might not work, or why a drug-like molecule does not structurally fit with a receptor. An ontology like MoSt may help with *explanations* and providing additional context for queries that are unanswered by machine learning algorithms. Furthermore, extending and mapping MoSt with SMARTS, MQL, and/or SLN might be a worthwhile avenue of interest as these three languages appear to be heavily used in querying chemical space. Since each query language provides various advantages, mapping MoSt to them might be beneficial to determine the limitations of the ontology's ability to represent (static, topological) chemical graphs.

Bibliography

[Aam16] Bahar Aameri. "Reasoning about Change with Domain-specific Process Ontologies". PhD thesis. University of Toronto, June 1, 2016 (cit. on pp. 136, 205).

[AG17] Bahar Aameri and Michael Grüninger. "Encountering the Physical World". In: *Proceedings of the Joint Ontology Workshops 2017 Episode 3: The Tyrolean Autumn of Ontology, Bozen-Bolzano, Italy, September 21-23, 2017.* 2017 (cit. on p. 11).

[AKM12] Saber A. Akhondi, Jan A. Kors, and Sorel Muresan. "Consistency of systematic chemical identifiers within and between small-molecule databases". In: *Journal of Cheminformatics* 4.1 (Dec. 2012), p. 35. ISSN: 1758-2946 (cit. on p. 18).

[AL95] Ron Aharoni and Martin Loebl. "Strongly perfect infinite graphs". In: *Israel Journal of Mathematics* 90.1 (Oct. 1995), pp. 81–91. ISSN: 1565-8511 (cit. on p. 174).

[Ame15] American Chemical Society. *Medicinal Chemistry*. 2015. URL: `http://www.acs.org/content/acs/en/careers/college-to-career/chemistry-careers/medicinal-chemistry.html` (cit. on p. 1).

[Bal85] Alexandru T. Balaban. "Applications of Graph Theory in Chemistry". In: *Journal of Chemical Information and Computer Sciences* 25.3 (1985), p. 335 (cit. on p. 12).

[Ben12] Mordechai Ben-Ari. "First-Order Logic: Undecidability and Model Theory *". In: *Mathematical Logic for Computer Science*. London: Springer London, 2012, pp. 223–230. ISBN: 978-1-4471-4129-7 (cit. on p. 202).

[BL04] Ronald Brachman and Hector Levesque. *Knowledge Representation and Reasoning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004. ISBN: 9780080489322 (cit. on pp. 222, 223).

[BLS99] A. Brandstädt, V.B. Le, and J.P. Spinrad. *Graph Classes: A Survey*. Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 1999. ISBN: 9780898714326 (cit. on pp. 162, 171).

[BM76] John Adrian Bondy and U.S.R. Murty. *Graph Theory With Applications*. Elsevier Science Ltd/North-Holland, 1976. ISBN: 9780444194510 (cit. on p. 142).

[BM96] Guy W. Bemis and Mark A. Murcko. "The properties of known drugs. 1. Molecular frameworks". In: *Journal of Medicinal Chemistry* 39.15 (1996), pp. 2887–2893 (cit. on pp. 13, 203).

[BM99] Guy W. Bemis and Mark A. Murcko. "Properties of Known Drugs. 2. Side Chains". In: *Journal of Medicinal Chemistry* 42.25 (1999), pp. 5095–5099 (cit. on p. 13).

[BMG96] 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 (cit. on pp. 187, 190).

[Bon91] Danail Bonchev. *Chemical graph theory: introduction and fundamentals*. Vol. 1. CRC Press, 1991 (cit. on p. 40).

[BP98] F. Buekenhout and M. Parker. "The Number of Nets of the Regular Convex Polytopes in Dimension ≤ 4". In: *Discrete Math.* 186.1-3 (May 1998), pp. 69–94. ISSN: 0012-365X (cit. on p. 10).

[BR12] R. Balakrishnan and K. Ranganathan. *A Textbook of Graph Theory (Universitext)*. Springer, 2012. ISBN: 978-1-4614-4528-9 (cit. on pp. 146, 159).

[Bro98] Frank K. Brown. "Chemoinformatics: What is it and How does it Impact Drug Discovery." In: *Annual Reports in Medicinal Chemistry*. Ed. by James A. Bristol. Vol. 33. Annual Reports in Medicinal Chemistry. Academic Press, 1998, pp. 375–384 (cit. on p. 13).

[Bru00] Robert Bruner. *What is Topology?* 2000. URL: `http://www.math.wayne.edu/~rrb/topology.html` (visited on 11/03/2016) (cit. on p. 2).

[BRW85] E. A. Bender, L. B. Richmond, and N. C. Wormald. "Almost all chordal graphs split". In: *Journal of the Australian Mathematical Society. Series A. Pure Mathematics and Statistics* 38.2 (1985), pp. 214–221 (cit. on p. 174).

[Bue95] Francis Buekenhout. *Handbook of Incidence Geometry: Buildings and Foundations*. North Holland, 1995. ISBN: 9780444883551 (cit. on p. 137).

[BW10] Robin S. Bon and Herbert Waldmann. "Bioactivity-Guided Navigation of Chemical Space". In: *Accounts of Chemical Research* 43.8 (2010). PMID: 20481515, pp. 1103–1114 (cit. on p. 187).

[BZÖ15] Sándor Benyhe, Ferenc Zádor, and Ferenc Ötvös. "Biochemistry of opioid (morphine) receptors: Binding, structure and molecular modelling". English. In: *Acta Biologica Szegediensis* 59 (2015), pp. 17–37. ISSN: 1588-385X (cit. on p. 96).

[CD11] Leonid Chepelev and Michel Dumontier. "Chemical Entity Semantic Specification: Knowledge representation for efficient semantic cheminformatics and facile data integration". In: *Journal of Cheminformatics* 3.1 (2011), p. 20. ISSN: 1758-2946 (cit. on p. 19).

[CG14] Carmen Chui and Michael Grüninger. "Mathematical Foundations for Participation Ontologies". In: *Formal Ontology in Information Systems - Proceedings of the Eighth International Conference, FOIS 2014, September, 22-25, 2014, Rio de Janeiro, Brazil*. 2014, pp. 105–118 (cit. on p. 137).

[CG16] Carmen Chui and Michael Grüninger. "A Molecular Structure Ontology for Medicinal Chemistry". In: *Formal Ontology in Information Systems - Proceedings of the 9th International Conference, FOIS 2016, Annecy, France, July 6-9, 2016*. 2016, pp. 285–298 (cit. on pp. 8, 27, 36).

[CG17] Carmen Chui and Michael Grüninger. "Verification and Modularization of the DOLCE Upper Ontology". In: *Proceedings of the Joint Ontology Workshops 2017 Episode 3: The Tyrolean Autumn of Ontology, Bozen-Bolzano, Italy, September 21-23, 2017*. 2017 (cit. on p. 137).

[CGW12] Jonathan Clayden, Nick Greeves, and Stuart G. Warren. *Organic Chemistry*. 2ed. Oxford University Press, 2012. ISBN: 9780199270293 (cit. on p. 14).

[Cha17] Charles River Laboratories, Inc. *The Benefits of Outsourcing Drug Discovery to an End-to-End CRO*. May 1, 2017. URL: `https://www.criver.com/resources/benefits-outsourcing-drug-discovery-end-end-cro` (cit. on p. 4).

[Che06] William Lingran Chen. "Chemoinformatics: Past, Present, and Future". In: *Journal of Chemical Information and Modeling* 46.6 (2006), pp. 2230–2255 (cit. on p. 13).

[Chu+13] Nitishal Chungoora et al. "A model-driven ontology approach for manufacturing system interoperability and knowledge sharing". In: *Computers in Industry* 64.4 (2013), pp. 392–401 (cit. on p. 2).

[CK116] CK-12 Foundation. *Solid Figures: Polyhedrons*. Aug. 25, 2016. URL: `https://www.ck12.org/geometry/solid-figures/lesson/Polyhedrons-BSC-GEOM/` (cit. on p. 9).

[Coh01] Anthony G. Cohn. "Formalising Bio-spatial Knowledge". In: *Proceedings of the International Conference on Formal Ontology in Information Systems - Volume 2001*. FOIS '01. Ogunquit, Maine, USA: ACM, 2001, pp. 198–209. ISBN: 1-58113-377-4 (cit. on pp. 23, 36, 119, 200).

[Cor+09] Thomas H. Cormen et al. *Introduction to Algorithms*. 3rd ed. The MIT Press, 2009. ISBN: 9780262033848 (cit. on p. 178).

[Cor86] D.G. Corneil. "Families of graphs complete for the strong perfect graph Conjecture". In: *Journal of Graph Theory* 10.1 (1986), pp. 33–40 (cit. on p. 163).

[Day07] Daylight Chemical Information Systems, Inc. *SMARTS - A Language for Describing Molecular Patterns*. 2007. URL: `http://www.daylight.com/dayhtml/doc/theory/theory.smarts.html` (cit. on p. 188).

[Deg+08] Kirill Degtyarenko et al. "ChEBI: a database and ontology for chemical entities of biological interest". In: *Nucleic Acids Research* 36.Database-Issue (2008), pp. 344–350 (cit. on pp. 13, 19).

[Dem+10] Emek Demir et al. "The BioPAX community standard for pathway data sharing". In: *Nature Biotechnology* 28.9 (Sept. 2010), pp. 935–942. ISSN: 1087-0156 (cit. on p. 2).

[Die12] Reinhard Diestel. *Graph Theory, 4th Edition*. Vol. 173. Graduate Texts in Mathematics. Springer, 2012. ISBN: 978-3-642-14278-9 (cit. on pp. 39, 41, 98, 99, 138, 139, 140, 141, 142).

[DPV11] Sanjoy Dasgupta, Christos Papadimitriou, and Umesh Vazirani. *Algorithms*. McGraw-Hill Education, 2011. ISBN: 9780073523408 (cit. on p. 98).

[DV09] Michael Dumontier and Natalia Villanueva-Rosales. "Towards pharmacogenomics knowledge discovery with the semantic web". In: *Briefings in Bioinformatics* 10.2 (2009), pp. 153–163 (cit. on p. 19).

[Ehr13] Hans-Christian Ehrlich. "Searching for Generic Chemical Patterns in Combinatorial Chemical Spaces". PhD thesis. University of Hamburg, Aug. 28, 2013 (cit. on p. 188).

[End72] Herbert B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 1972, pp. I–XIII, 1–295. ISBN: 978-0-12-238450-9 (cit. on pp. 115, 119, 121, 223, 226).

[FD10] Fdardel and DMacks. *Deriving the SMILES representation of a chemical molecule, Shown example: ciprofloxacin, a fluoroquinolone antibiotic.* June 27, 2010. URL: `https://commons.wikimedia.org/wiki/File:SMILES.png` (cit. on p. 16).

[FDF74] "Origin and Evolution of Organic Nomenclature". In: *Nomenclature of Organic Compounds*. Ed. by John H. Fletcher, Otis C. Dermer, and Robert B. Fox. Vol. 126. American Chemical Society, 1974. Chap. 2, pp. 1–5 (cit. on pp. 13, 184).

[Fre+16] Scott Freeman et al. *Biological Science*. 6th ed. Pearson, 2016. ISBN: 9780321976499 (cit. on pp. 57, 60, 63, 64).

[GB11] Michael Grüninger and Salim Bouafoud. "Thinking Outside (and Inside) the Box". In: *Proceedings of the First Interdisciplinary Workshop on SHAPES, Karlsruhe, Germany, September 27, 2011*. 2011 (cit. on pp. 1, 11, 205).

[GD09] Michael Grüninger and Arnaud Delaval. "A First-Order Cutting Process Ontology for Sheet Metal Parts". In: *Formal Ontologies Meet Industry, Proceedings of the 4th Workshop FOMI 2009, September 2, 2009, Vicenza, Italy, in association with the 10th European Conference on Knowledge Management*. 2009, pp. 22–33 (cit. on p. 11).

[GHK11] Michael Grüninger, Torsten Hahmann, and Megan Katsumi. "Exploiting Modularity for Ontology Verification". In: *Modular Ontologies - Proceedings of the Fifth International Workshop, WoMO 2011, Ljubljana, Slovenia, August 2011*. 2011, pp. 55–62 (cit. on p. 137).

[GHM08] Michael Grüninger, Richard Hull, and Sheila A. McIlraith. "A Short Overview of FLOWS: A First-Order Logic Ontology for Web Services". In: *IEEE Data Eng. Bull.* 31.3 (2008), pp. 3–7 (cit. on p. 2).

[GM03] Michael Grüninger and Christopher Menzel. "The Process Specification Language (PSL) Theory and Applications". In: *AI Mag.* 24.3 (Sept. 2003), pp. 63–74. ISSN: 0738-4602 (cit. on pp. 2, 117).

[GO11] Michael Grüninger and Darren Ong. "Verification of Time Ontologies with Points and Intervals". In: *TIME*. 2011, pp. 31–38 (cit. on pp. 120, 137).

[Göd92] Kurt Gödel. *On Formally Undecidable Propositions of Principia Mathematica and Related Systems*. Trans. by B. Meltzer. Dover Publications, Apr. 1, 1992. ISBN: 0486669807 (cit. on p. 24).

[GOS09] Nicola Guarino, Daniel Oberle, and Steffen Staab. "What is an ontology?" In: *Handbook on ontologies*. Springer, 2009, pp. 1–17 (cit. on pp. 31, 120, 136).

[Gri10] Ivan Griffin. *Periodic Table of Chemical Elements in TikZ*. 2010. URL: `http://www.texample.net/tikz/examples/periodic-table-of-chemical-elements/` (cit. on p. 46).

[Grü+10] Michael Grüninger et al. "Ontology Verification with Repositories". In: *FOIS*. 2010, pp. 317–330 (cit. on pp. 28, 120, 121, 135, 137, 160).

[Grü+12] Michael Grüninger et al. "Modular First-Order Ontologies via Repositories". In: *Applied Ontology* 7.2 (2012), pp. 169–209 (cit. on pp. 28, 117, 118, 119, 120, 121, 122).

[Grü00] Michael Grüninger. *Logical foundations of shape-based object recognition*. PhD thesis. University of Toronto, 2000 (cit. on p. 11).

[Grü11] Michael Grüninger. "Verification of the OWL-time Ontology". In: *Proceedings of the 10th International Conference on The Semantic Web - Volume Part I*. ISWC'11. Bonn, Germany: Springer-Verlag, 2011, pp. 225–240. ISBN: 978-3-642-25072-9 (cit. on pp. 28, 29, 102, 135, 137).

[Grü18] Michael Grüninger. *Incidence Structures for Applied Ontology*. Tech. rep. 2018 (cit. on pp. 147, 148, 153).

[Gru95] Thomas R. Gruber. "Toward Principles for the Design of Ontologies Used for Knowledge Sharing". In: *Int. J. Hum.-Comput. Stud.* 43.5-6 (1995), pp. 907–928 (cit. on p. 2).

[Grü99] *Papers from the AAAI Workshop. Adam Farquhar and Kilian Stoffel, Cochairs. Semantic Characterization of Ontologies. Ontology Management*. Technical Report WS-99-13. Menlo Park, California, 1999, pp. 28–33 (cit. on p. 117).

[Gün88] Maier Günther. "Tetrahedrane and Cyclobutadiene". In: *Angewandte Chemie International Edition in English* 27.3 (Mar. 1988), pp. 309–332 (cit. on p. 185).

[Has+10] Janna Hastings et al. "What are chemical structures and their relations?" In: *Formal Ontology in Information Systems, Proceedings of the Sixth International Conference, FOIS 2010, Toronto, Canada, May 11-14, 2010*. 2010, pp. 257–270 (cit. on p. 21).

[Has+11] Janna Hastings et al. "The Chemical Information Ontology: Provenance & Disambiguation for Chemical Data on the Biological Semantic Web". In: *PLoS ONE* 6.10 (2011), e25513 (cit. on p. 13).

[Has+13] Janna Hastings et al. "The ChEBI reference database and ontology for biologically relevant chemistry: enhancements for 2013". In: *Nucleic Acids Research* 41.D1 (2013), pp. D456–D463 (cit. on pp. 19, 21).

[Has08] Ali Hashemi. "Using Repositories for Ontology Design and Semantic Mapping". MA thesis. Toronto, Canada: University of Toronto, Department of Mechanical and Industrial Engineering, 2008. ISBN: 9780494589342 (cit. on p. 27).

[Hay96] Patrick Hayes. *A Catalog of Temporal Theories*. Tech. rep. UIUC-BI-AI-96-01. Beckman Institute, Departments of Philosophy, and Computer Science, University of Illinois, 1996 (cit. on pp. 2, 120).

[HBO13] Janna Hastings, Colin R. Batchelor, and Mitsuhiro Okada. "Shape Perception in Chemistry". In: *SHAPES*. 2013, pp. 83–94 (cit. on p. 1).

[HBS11] Janna Hastings, Colin R. Batchelor, and Stefan Schulz. "Parts and Wholes, Shapes and Holes in Living Beings". In: *Proceedings of the First Interdisciplinary Workshop on SHAPES, Karlsruhe, Germany, September 27, 2011*. 2011 (cit. on p. 1).

[Hel+15] Stephen Heller et al. "InChI, the IUPAC International Chemical Identifier". In: *J. Cheminform.* 7.1 (2015), p. 23. ISSN: 1758-2946 (cit. on p. 17).

[HKM11] Janna Hastings, Oliver Kutz, and Till Mossakowski. "How to model the shapes of molecules? Combining topology and ontology using heterogeneous specifications". In: *Deep Knowledge Representation Challenge Workshop, co-located with K-CAP 2011*. 2011 (cit. on pp. 23, 24, 25, 200).

[Hom+08] R. Webster Homer et al. "SYBYL Line Notation (SLN): A Single Notation To Represent Chemical Structures, Queries, Reactions, and Virtual Libraries". In: *Journal of Chemical Information and Modeling* 48.12 (2008). PMID: 18998666, pp. 2294–2307 (cit. on p. 190).

[Hor+11] Eelke van der Horst et al. "Chemogenomics Approaches for Receptor Deorphanization and Extensions of the Chemogenomics Concept to Phenotypic Space". In: *Current Topics in Medicinal Chemistry* (Aug. 2011), pp. 1964–1977. ISSN: 1568-0266 (cit. on p. 187).

[HP04] Jerry R. Hobbs and Feng Pan. "An Ontology of Time for the Semantic Web". In: 3.1 (Mar. 2004), pp. 66–85. ISSN: 1530-0226 (cit. on p. 2).

[Hun14] Ian Hunt. *Basic IUPAC Organic Nomenclature*. 2014. URL: `http://www.chem.ucalgary.ca/courses/351/WebContent/orgnom/main/rules.html` (cit. on p. 218).

[Hun16] Edward V. Huntington. "A Set of Independent Postulates for Cyclic Order". In: *Proceedings of the National Academy of Sciences* 2.11 (1916), pp. 630–631. ISSN: 0027-8424 (cit. on p. 100).

[InC18] InChI Trust. *Technical FAQ - 2.3. What is the purpose of the InChI?* 2018. URL: `https://www.inchi-trust.org/technical-faq/#2.3` (visited on 01/01/2018) (cit. on pp. 17, 18).

[Int07] International Organization for Standardization. *ISO 24707:2007: Information technology - Common Logic (CL): A Framework for a Family of Logic-based Languages*. Norm. 2007 (cit. on p. 27).

[IUP06] IUPAC. *Compendium of Chemical Terminology, 2nd ed. (the "Gold Book"). Compiled by A. D. McNaught and A. Wilkinson. Blackwell Scientific Publications, Oxford (1997). XML on-line corrected version: http://goldbook.iupac.org (2006–) created by M. Nic, J. Jirat, B. Kosata; updates compiled by A. Jenkins*. 2006. ISBN: ISBN 0-9678550-9-8 (cit. on pp. 55, 106).

[Jam16] Craig A. James. *OpenSMILES Specification*. May 15, 2016. URL: `http://opensmiles.org/opensmiles.html` (cit. on p. 17).

[Kat16] Megan Katsumi. "Principles and Practices of Ontology Reuse". PhD thesis. University of Toronto, Nov. 1, 2016 (cit. on p. 31).

[KDD08] Mykola Konyk, Alexander De Leon, and Michel Dumontier. "Chemical Knowledge for the Semantic Web". In: *Data Integration in the Life Sciences*. Ed. by Amos Bairoch, Sarah Cohen-Boulakia, and Christine Froidevaux. Vol. 5109. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, pp. 169–176 (cit. on p. 19).

[Koc+05] Marcus A. Koch et al. "Charting biologically relevant chemical space: A structural classification of natural products (SCONP)". In: *Proceedings of the National Academy of Sciences of the United States of America* 102.48 (2005), pp. 17272–17277 (cit. on p. 5).

[Lac+05] Lee W. Lacy et al. "Experiences with using OWL in Military Applications". In: *Proceedings of the OWLED'05 Workshop on OWL: Experiences and Directions*. Ed. by Bernardo Cuenca Grau et al. Vol. 188. Galway, Ireland: CEUR Workshop Proceedings (CEUR-WS.org), Nov. 11, 2005 (cit. on p. 2).

[Les+09] Uta Lessel et al. "Searching Fragment Spaces with Feature Trees". In: *Journal of Chemical Information and Modeling* 49.2 (2009). PMID: 19159249, pp. 270–279 (cit. on p. 191).

[Lew+98] 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). PMID: 9611787, pp. 511–522 (cit. on p. 191).

[Lew91] David Lewis. *Parts of Classes*. Vol. 43. 172. Basil Blackwell, Inc., July 1991, p. 362 (cit. on p. 113).

[Mag13] Despoina Magka. "Foundations and applications of knowledge representation for structured entities". PhD thesis. University of Oxford, 2013 (cit. on pp. 21, 22, 24, 36, 45, 47, 54).

[McM15] John E. McMurry. *Organic Chemistry*. 9th ed. Cengage Learning, 2015. ISBN: 9781305080485 (cit. on pp. 2, 60, 218).

[Meg76] Nimrod Megiddo. "Partial and complete cyclic orders". In: *Bull. Amer. Math. Soc.* 82.2 (Mar. 1976), pp. 274–276 (cit. on p. 102).

[MG16] Lydia Silva Muñoz and Michael Grüninger. "Locating Things in Space and Time: Verification of the SUMO Upper-Level Ontology". In: *Knowledge Engineering and Knowledge Management - 20th International Conference, EKAW 2016, Bologna, Italy, November 19-23, 2016, Proceedings*. 2016, pp. 606–620 (cit. on p. 137).

[MKH14] Despoina Magka, Markus Krotzsch, and Ian Horrocks. "A rule-based ontological framework for the classification of molecules". In: *Journal of Biomedical Semantics* 5.1 (2014), p. 17. ISSN: 2041-1480 (cit. on pp. 21, 22, 36, 45, 54, 119, 200).

[MOF14] Norberto K. V. Monteiro, José F. de Oliveira, and Caio L. Firme. "Stability and electronic structures of substituted tetrahedranes, silicon and germanium parents - a DFT, ADMP, QTAIM and GVB study". In: *New J. Chem.* 38 (12 2014), pp. 5892–5904 (cit. on p. 185).

[Moo17] G. Eric Moorhouse. *Incidence Geometry. Math 5700 Course Notes*. University of Wyoming. Aug. 1, 2017. URL: `http://ericmoorhouse.org/handouts/Incidence_Geometry.pdf` (cit. on p. 137).

[Mos99] G. P. Moss. "Extension and revision of the von Baeyer system for naming polycyclic compounds (including bicyclic compounds)". In: *Pure and Applied Chemistry* 71.3 (1999), pp. 513–529 (cit. on pp. 85, 86).

[MS02] Peter McMullen and Egon Schulte. *Abstract Regular Polytopes*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2002 (cit. on p. 10).

[MS07] Harald Mauser and Martin Stahl. "Chemical Fragment Spaces for de novo Design". In: *Journal of Chemical Information and Modeling* 47.2 (2007). PMID: 17300171, pp. 318–324 (cit. on p. 191).

[Mun00] J.R. Munkres. *Topology*. Featured Titles for Topology Series. Prentice Hall, Incorporated, 2000. ISBN: 9780131816299 (cit. on p. 2).

[Mus01] Frank J. Mustoe. *McGraw-Hill Ryerson Chemistry 11*. Toronto: McGraw-Hill Ryerson, 2001. ISBN: 9780070886810 (cit. on p. 45).

[Mus02] Frank J. Mustoe. *McGraw-Hill Ryerson Chemistry 12*. Toronto: McGraw-Hill Ryerson, Aug. 21, 2002. ISBN: 9780070916432 (cit. on pp. 45, 60).

[Nat15] National Center for Biotechnology Information. *PubChem Compound Database - Morphine (CID=5288826)*. 2015. URL: `https://pubchem.ncbi.nlm.nih.gov/compound/5288826` (cit. on p. 18).

[Neb07] Ladislav Nebeský. "A new approach to chordal graphs". In: *Czechoslovak Mathematical Journal* 57.1 (Mar. 2007), pp. 465–471. ISSN: 1572-9141 (cit. on pp. 174, 175, 176).

[OBo12] Noel M. O'Boyle. "Towards a Universal SMILES representation - A standard method to generate canonical SMILES based on the InChI". In: *Journal of Cheminformatics* 4.1 (Sept. 2012), p. 22. ISSN: 1758-2946 (cit. on p. 18).

[Opr05] Tudor I. Oprea. "Chemoinformatics in Lead Discovery". In: *Chemoinformatics in Drug Discovery*. Wiley-VCH Verlag GmbH & Co. KGaA, 2005. Chap. 2, pp. 23–41. ISBN: 9783527603749 (cit. on p. 13).

[Pat13] Graham L. Patrick. *An Introduction to Medicinal Chemistry*. 5th ed. Oxford University Press, Jan. 18, 2013. ISBN: 9780199697397 (cit. on p. 203).

[PM17] David L. Poole and Alan K. Mackworth. *Artificial Intelligence: Foundations of Computational Agents*. 2nd ed. New York, NY, USA: Cambridge University Press, 2017, p. 820. ISBN: 978-1107195394 (cit. on p. 178).

[Pro+07] Ewgenij Proschak et al. "Molecular Query Language (MQL) - A Context-Free Grammar for Substructure Matching". In: *Journal of Chemical Information and Modeling* 47.2 (2007). PMID: 17381167, pp. 295–301 (cit. on pp. 188, 190).

[Pur+93] International Union of Pure et al. *A Guide to IUPAC Nomenclature of Organic Compounds: Recommendations 1993*. IUPAC chemical data series. Blackwell Scientific Publications, 1993. ISBN: 9780632037025 (cit. on pp. 85, 86, 184).

[RA12] Jean-Louis Reymond and Mahendra Awale. "Exploring Chemical Space for Drug Discovery Using the Chemical Universe Database". In: *ACS Chemical Neuroscience* 3.9 (2012). PMID: 23019491, pp. 649–657 (cit. on pp. 4, 187, 188, 189).

[Rov11] Robert J. Rovetto. "The Shape of Shapes: An Ontological Exploration". In: *Proceedings of the First Interdisciplinary Workshop on SHAPES, Karlsruhe, Germany, September 27, 2011*. 2011 (cit. on p. 1).

[RP05] Robert G. Raskin and Michael J. Pan. "Knowledge representation in the semantic web for Earth and environmental terminology (SWEET)". In: *Computers & Geosciences* 31.9 (2005). Application of XML in the Geosciences, pp. 1119–1125. ISSN: 0098-3004 (cit. on p. 2).

[Sch+07] Ansgar Schuffenhauer et al. "The Scaffold Tree - Visualization of the Scaffold Universe by Hierarchical Scaffold Classification". In: *Journal of Chemical Information and Modeling* 47.1 (2007). PMID: 17238248, pp. 47–58 (cit. on pp. 4, 203, 204).

[Ser15] Robert F. Service. "The synthesis machine". In: *Science* 347.6227 (2015), pp. 1190–1193 (cit. on p. 26).

[Sha91] Stewart Shapiro. *Foundations without Foundationalism: A Case for Second-Order Logic (Oxford Logic Guides)*. Oxford University Press, 1991. ISBN: 9780198533917 (cit. on p. 226).

[STW12] Hongmao Sun, Gregory Tawa, and Anders Wallqvist. "Classification of Scaffold Hopping Approaches". In: *Drug Discovery Today* 17.7-8 (Oct. 2012), pp. 310–324 (cit. on pp. 4, 203).

[Szc77] L.W. Szczerba. "Interpretability of Elementary Theories". English. In: *Logic, Foundations of Mathematics, and Computability Theory*. Ed. by RobertE. Butts and Jaakko Hintikka. Vol. 9. The University of Western Ontario Series in Philosophy of Science. Springer Netherlands, 1977, pp. 129–145. ISBN: 978-94-010-1140-2 (cit. on p. 122).

[TC09] Roberto Todeschini and Viviana Consonni. *Molecular Descriptors for Chemoinformatics*. Jan. 2009 (cit. on p. 187).

[The17] The LibreTexts Libraries. *Microbiology (Boundless). 2.5.4: Amino Acids*. Dec. 8, 2017. URL: https://bio.libretexts.org/TextMaps/Microbiology/Book%3A_Microbiology_(Boundless)/2%3A_Chemistry/2.5%3A_Organic_Compounds/2.5.4%3A_Amino_Acids (visited on 12/08/2017) (cit. on p. 58).

[Tri92] Nenad Trinajstić. *Chemical Graph Theory*. Boca Raton, Ann Arbor, London: CRC Press, Inc, 1992. ISBN: 0-8493-4256-2 (cit. on pp. 12, 36, 40, 119).

[Var16] Achille Varzi. *Mereology. The Stanford Encyclopedia of Philosophy (Winter 2016 Edition)*. Ed. by Edward N. Zalta. Feb. 13, 2016. URL: https://plato.stanford.edu/entries/mereology/#OthMerCon (cit. on p. 113).

[VD07] Natalia Villanueva-Rosales and Michel Dumontier. "Describing Chemical Functional Groups in OWL-DL for the Classification of Chemical Compounds". In: *Proceedings of the OWLED 2007 Workshop on OWL: Experiences and Directions, Innsbruck, Austria, June 6-7, 2007*. 2007 (cit. on p. 19).

[WD15] William Weiss and Cherie D'Mello. *Fundamentals of Model Theory*. 2015 (cit. on p. 198).

[Wei88] David Weininger. "SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules". In: *Journal of Chemical Information and Computer Sciences* 28.1 (1988), pp. 31–36 (cit. on p. 15).

[Wil10] William McCune. *Prover9 and Mace4*. 2010. URL: `http://www.cs.unm.edu/~mccune/prover9/` (cit. on p. 224).

[WWW89] David Weininger, Arthur Weininger, and Joseph L. Weininger. "SMILES. 2. Algorithm for generation of unique SMILES notation". In: *Journal of Chemical Information and Computer Sciences* 29.2 (1989), pp. 97–101 (cit. on p. 17).

[Yuz+13] Kobayashi Yuzuru et al. "Cross-Coupling Reaction of a Highly Strained Molecule: Synthesis of σ-π Conjugated Tetrahedranes". In: *Angewandte Chemie International Edition* 52.41 (Oct. 2013), pp. 10740–10744 (cit. on p. 185).

Appendix A

Supplementary Chemistry Material

A.1 IUPAC Nomenclature

The IUPAC systematic naming of an organic compound is based on a series of steps and rules [Hun14; McM15]:

1. **Identification of the principal functional group and substituents**: Functional groups are divided into two classes, principal groups and subordinate groups, as seen in Table A.1. Principal groups can be cited as prefixes or suffixes, whereas subordinate groups are cited only as prefixes. The proper suffix for a given compound is determined by choosing the principal group fo the highest priority.

2. **Identification of the longest continuous chain containing the principal functional group**: If the principal group of highest priority is part of an open chain, the parent name is of the longest chain with the largest number of principal groups. If the highest-priority group is attached to a ring, then the parent is the name of the ring system.

3. **Numbering and naming the principal functional group and substituents**: The next step is to identify and to give numbers (locants) to all substituents on the parent chain or ring. Substituents include all akyl groups and functional groups other than the one cited in the suffix.

With the components of the molecule established, there are several additional rules that apply with writing out names of molecules according to IUPAC:

1. **Order of prefixes**: the name is written with the substituents listed in alphabetical order, rather than numerical.

2. **Using hyphens for single- and multiple-word names**: if the parent is an element or compound, then the name is written as a single world. If not, then the name is written in multiple words.

3. **Parentheses**: parentheses are used to denote complex substituents to prevent ambiguity.

Additional explanations of the IUPAC nomenclature rules can be found in the online version of the Blue Book: `https://www.acdlabs.com/iupac/nomenclature/`

Table A.1: Classification of Functional Groups. Principal groups are listed in order of decreasing priority, while subordinate groups have no priority.

Principal Functional Group	**Name As Suffix**	**Name As Prefix**
Carboxylic Acids	-oic acid -carboxylic acid	carboxy
Acid anhydries	-oic anhydride -carboxylic anhydride	–
Esters	-oate -carboxylate	alkoxycarbonyl
Thioesters	-thioate -carbothioate	alkylthiocarbonyl
Acid Halides	-oyl halide -carbonyl halide	halocarbonyl
Amides	-amide -carboxamide	carbamoyl
Nitriles	-nitrile -carbonitrile	cyano
Aldehydes	-al -carbaldehyde	oxo
Ketones	-one	oxo
Alcohols	-ol	hydroxy
Phenols	-ol	hydroxy
Thiols	-thiol	mercapto
Amines	-amine	amino
Imines	-imine	imino
Ethers	ether	alkoxy
Sulfides	sulfide	akylthio
Disulfides	disulfide	–
Alkenes	-ene	–
Alkynes	-yne	–
Alkanes	-ane	–
Subordinate Functional Group	**Name As Suffix**	**Name As Prefix**
Azides	–	azido
Halides	–	halo
Nitro compounds	–	nitro

A.2 Hydrocarbons

Tables A.2 and A.3 outline the names for the hydrocarbon framework in organic molecules.

Table A.2: Naming Hydrocarbons

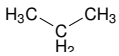
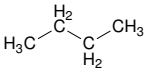
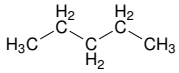
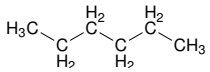
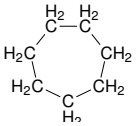
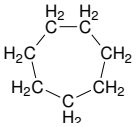
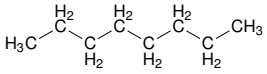
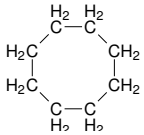
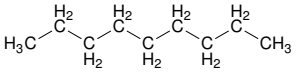
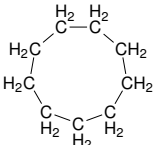
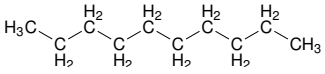
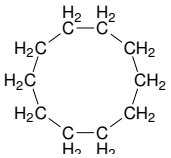
# Carbons	Compound	Structural Diagram
1	methane	
2	ethane	
3	propane	
	cyclopropane	
4	butane	
	cyclobutane	
5	pentane	
	cyclopentane	

Table A.3: Naming Hydrocarbons (cont.)

# Carbons	Compound	Structural Diagram
6	hexane	
	cyclohexane	
7	heptane	
	cycloheptane	
8	octane	
	cyclooctane	
9	nonane	
	cyclononane	
10	decane	
	cyclodecane	

Appendix B

Supplementary Logic Material

B.1 First-Order Logic

In first-order logic, there are two types of symbols: logical and non-logical ones. The following tables, adapted from [BL04], summarize and explain the differences of each.

Table B.1: Logical syntax in first-order logic.

Punctuation	"(", ")", and "."
Connectives	"¬" Logical negation
	"∧" Logical conjunction
	"∨" Logical disjunction
	"∃" Existential quantifier ("there exists")
	"∀" Universal quantifier ("for all")
	"≡" Logical equality
Variables	We will denote these using x, y, and z, sometimes with subscripts and superscripts.

Table B.2: Non-Logical Symbols in first-order logic.

Function Symbols	We will write in uncapitalized mixed case (e.g., $bestFriend$) and which will be denoted more generally using a, b, c, f, g and h, with subscripts and superscripts.
Predicate Symbols	We will write in uncapitalized mixed case (e.g. $olderThan$) and which will be denoted more generally using P, Q, and R, with subscripts and sueprscripts.

B.1.1 Operator Precedence

Precedence among logical operators is defined as follows (left → right):

$$\neg,\ \wedge,\ \vee,\ \supset,\ \equiv$$

Extra parentheses may be inserted to make sentences easier to read.

For example, consider the following:

$$\forall x\ P(x) \supset \exists y \exists z\ Q(y,z) \wedge \neg \exists x\ R(x)$$

can be also written as

$$(\forall x\ P(x)) \supset \exists y (\exists z\ (Q(y,z) \wedge \neg (\exists x\ (R(x)))$$

Note: the *inner* occurrence of x is bound to the *innermost* existential quantifier for $R(x)$.

B.1.2 Identities & Logical Relationships

Propositional Identities

$$\neg(P \vee Q) \Leftrightarrow (\neg P \wedge \neg Q)$$
$$\neg(P \wedge Q) \Leftrightarrow (\neg P \vee \neg Q)$$
$$[P \wedge (Q \vee R)] \Leftrightarrow [(P \wedge Q) \vee (P \wedge R)]$$
$$[P \vee (Q \wedge R)] \Leftrightarrow [(P \vee Q) \wedge (P \vee R)]$$
$$(P \supset Q) \Leftrightarrow (\neg P \vee Q)$$
$$\neg(P \supset Q) \Leftrightarrow (P \wedge \neg Q)$$
$$[(P \wedge Q) \supset R)] \Leftrightarrow [P \supset (Q \supset R)]$$

Logical Relationships Involving Quantifiers

$$(\neg(\forall x)\ p(x)) \Leftrightarrow (\exists x) \neg p(x)$$
$$(\neg(\exists x)\ p(x)) \Leftrightarrow (\forall x) \neg p(x)$$

B.1.3 Properties of Relations

Table B.3: Properties of logical relations.

A relation R is...	**...if...**
Reflexive	$\forall x\ R(x,x)$
Irreflexive	$\forall x\ \neg R(x,x)$
Transitive	$\forall x \forall y \forall z\ [(R(x,y) \wedge R(y,z)) \supset R(x,z)]$
Symmetric	$\forall x \forall y\ (R(x,y) \supset R(y,x))$
Asymmetric	$\forall x \forall y\ (R(x,y) \supset \neg R(y,x))$
Antisymmetric	$\forall x \forall y\ ((R(x,y) \wedge R(y,x)) \supset (x = y))$

For more information on first-order logic, please see Chapter 2 in [BL04] and Chapter 2 in [End72].

B.2 Prover9 Syntax

Prover9 is an automated theorem prover for first-order and equational logic, and is the successor of the Otter theorem prover [Wil10]. Prover9 takes in input files that contain lists of clauses to determine whether a refutation exists by utilizing the resolution inference rule. In order to prove sentences in Prover9, one must specify the parameters of its inference procedure, the sentence to be proved (also known as the 'goal'), and the sentences that can be used in the proof.

The Prover9 syntax is similar to first-order logic:

In English	If x is the mother of y, then x is a parent of y.
In first-order logic	$\forall x \forall y\,(mother(x,y) \supset parent(x,y))$
Prover9 Syntax	`-mother	parent(x,y).`

Figure B.1: FOL and Prover9 Syntax

B.3 Common Logic

Table B.4: Overview of the first-order logic, Prover9 and Common Logic syntaxes.

Symbol/Meaning	FOL	Prover9	Common Logic
universal quantifier	$\forall$	all	forall
existential quantifier	$\exists$	exists	exists
conjunction	$\wedge$	&	and
disjunction	$\vee$	|	or
negation	$\neg$	-	not
implication	p(x) $\supset$ q(x)	p(x) ->q(x)	(if (p x)(q x))
equivalence	p(x) $\equiv$ q(x)	p(x)<->q(x)	(iff (p x)(q x))

B.4 Examples of Sentences in FOL, Prover9 and CLIF

1. "If x is a parent of y, then x is an ancestor of y."

   - First-order logic syntax:

     $$(\forall x \forall y(parent(x,y) \supset ancestor(x,y)))$$

   - Prover9 syntax:

     ```
     (all x all y (parent(x,y) -> ancestor(x,y))).
     ```

   - Common Logic syntax:

```
(forall (x y)
(if (parent x y)
(ancestor x y)))
```

2. "There exists someone who is not the mother of anyone else."

   - First-order logic syntax:

   $$\exists x(\forall y(\neg mother(x, y)))$$

   - Prover9 syntax:

   ```
   (exists x (all y (-mother(x,y)))).
   ```

   - Common Logic syntax:

   ```
   (exists (x)
   (forall (y)
   (not (mother x y))))
   ```

3. "The parent of someone is either their mother or father."

   - First-order logic syntax:

   $$(\forall x \forall y(parent(x, y) \supset mother(x, y) \vee father(x, y)))$$

   - Prover9 syntax:

   ```
   (all x all y (parent(x,y) ->
   (mother(x,y) | father(x,y)))).
   ```

   - Common Logic syntax:

   ```
   (forall (x y)
   (if (parent x y)
   (or (mother x y)(father x y))))
   ```

4. "Every cloud has a silver lining."

   - First-order logic syntax:

   $$\forall x(cloud(x) \supset (\exists y(lining(y) \wedge silver(y) \wedge has(x, y)))$$

   - Prover9 syntax:

   ```
   all x (cloud(x) ->
   (exists y (lining(y) & silver(y) & has(x,y)))).
   ```

   - Common Logic syntax:

```
(forall (x)
(if (cloud x)
(exists (y)
(and (lining y) (silver y) (has x y))))
```

5. "If t_1 and t_2 are time points, then t_1 is before t_2, t_2 is before t_1, or they are the same time point."

   - First-order logic syntax:

   $$\forall t_1 \forall t_2((timepoint(t_1) \wedge timepoint(t_2)) \supset$$

   $$(before(t_1, t_2) \vee before(t_2, t_1) \vee (t_1 = t_2)))$$

   - Prover9 syntax:

   ```
   all t1 all t2 ((timepoint(t1) & timepoint(t2)) ->
   (before(t1,t2) | before(t2,t1) | (t1 = t2))).
   ```

   - Common Logic syntax:

   ```
   (forall (t1 t2)
   (if (and (timepoint t1)(timepoint t2))
   (or (before t1 t2)(before t2 t1)(= t1 t2))))
   ```

B.5 Second-Order Logic

In first-order logic, we quantify over individuals, but not properties. For example, an atomic sentence like $Bear(b)$ can be quantified with a variable and a quantifier:

$$\exists x\, Bear(x)$$

But this cannot be done with the predicate in first-order logic:

$$\exists P\, P(b)$$

This is, however, a legitimate sentence in second-order logic, which has more "expressive power." For example, in first-order logic, it would not be possible to say that a and b have some property in common, but it can be expressed in second-order logic as:

$$\exists P\, (P(a) \wedge P(b))$$

In *second-order logic*, variables can be quantified over both terms and relations.

For more information on second-order logic, please see [Sha91] and Chapter 4 in [End72].

Appendix C

Other Axioms & Models

C.1 Axioms of $T_{cycle\_path\_subgraph\_nonisolated}$

The axioms of $T_{cycle\_path\_subgraph\_nonisolated}$ can be found in COLORE[1]. The following axioms were not included in the verification of T_{most_group}:

$$(\forall p \forall q\ (star(p,q) \equiv\ (\exists l_1 \exists l_2 \exists l_3 (line(l_1) \wedge line(l_2) \wedge line(l_3) \wedge l_1 \neq l_2 \wedge l_1 \neq l_3 \wedge l_2 \neq l_3 \wedge in(p,l_1) \wedge in(p,l_2) \wedge in(p,l_3) \wedge in(l_1,q) \wedge in(l_2,q) \wedge in(l_3,q))))). \quad \text{(CPSN-19)}$$

$$(\forall l \forall q\ (hanging(l,q) \equiv line(l) \wedge plane(q) \wedge (\exists p_1 (point(p_1) \wedge in(p_1,q) \wedge in(p_1,l) \wedge (\forall p_2 (point(p_2) \wedge in(p_2,l) \wedge in(p_2,q) \supset p_1 = p_2)))))). \quad \text{(CPSN-20)}$$

$$(\forall x \forall q\ (planar\_pendant(x,q) \equiv point(x) \wedge plane(q) \wedge in(x,q) \wedge (\forall y \forall z (plane\_collinear(x,y,q) \wedge plane\_collinear(x,z,q) \supset y = z)))). \quad \text{(CPSN-21)}$$

$$(\forall l\ (border\_line(l) \equiv (\exists q (line(l) \wedge plane(q) \wedge in(l,q) \wedge (\forall q_2 (plane(q_2) \wedge in(l,q_2) \supset q_2 = q)))))). \quad \text{(CPSN-22)}$$

$$(\forall p_1 \forall p_2 \forall q\ (plane\_collinear(p_1,p_2,q) \equiv point(p_1) \wedge point(p_2) \wedge p_1 \neq p_2 \wedge plane(q) \wedge in(p_1,q) \wedge in(p_2,q) \wedge (\exists l (line(l) \wedge in(l,q) \wedge in(p_1,l) \wedge in(p_2,l)))))). \quad \text{(CPSN-23)}$$

$$(\forall q\ (matching(q) \equiv plane(q) \wedge (\forall l_1 \forall l_2 (line(l_1) \wedge line(l_2) \wedge l_1 \neq l_2 \wedge in(l_1,q) \wedge in(l_2,q) \supset parallel(l_1,l_2))))). \quad \text{(CPSN-24)}$$

[1] http://colore.oor.net/tripartite_incidence/cycle_path_subgraph_nonisolated.clif

$$(\forall p \forall q_1 \forall q_2\ (junction(p,q_1,q_2) \equiv point(p) \wedge plane(q_1) \wedge plane(q_2) \wedge q_1 \neq q_2 \wedge (\exists l \exists l_1 \exists l_2 (line(l) \wedge line(l_1) \wedge line(l_2) \wedge in(p,l) \wedge in(p,l_1) \wedge in(p,l_2) \wedge in(l,q_1) \wedge in(l,q_2) \wedge in(l_1,q_1) \wedge \neg in(l_1,q_2) \wedge \neg in(l_2,q_1) \wedge in(l_2,q_2))))). \quad \text{(CPSN-25)}$$

$$(\forall p_1 \forall p_2\ (collinear(p_1,p_2) \equiv point(p_1) \wedge point(p_2) \wedge p_1 \neq p_2 \wedge (\exists l (line(l) \wedge in(p_1,l) \wedge in(p_2,l))))). \quad \text{(CPSN-26)}$$

$$(\forall l_1 \forall l_2\ (intersect(l_1,l_2) \equiv line(l_1) \wedge line(l_2) \wedge l_1 \neq l_2 \wedge (\exists p (point(p) \wedge in(p,l_1) \wedge in(p,l_2))))). \quad \text{(CPSN-27)}$$

$$(\forall l_1 \forall l_2\ (parallel(l_1,l_2) \equiv line(l_1) \wedge line(l_2) \wedge \neg(\exists p (point(p) \wedge in(p,l_1) \wedge in(p,l_2))))). \quad \text{(CPSN-28)}$$

$$(\forall l_1 \forall l_2\ (nontrivial\_segment(l_1,l_2) \equiv (\exists p_1 (point(p_1) \wedge in(p_1,l_1))) \wedge (\forall p_2 (point(p_2) \wedge in(p_2,l_1) \supset in(p_2,l_2))))). \quad \text{(CPSN-29)}$$

$$(\forall l_1 \forall l_2\ (adj(l_1,l_2) \equiv line(l_1) \wedge line(l_2) \wedge (\exists p (point(p) \wedge in(p,l_1) \wedge in(p,l_2))))). \quad \text{(CPSN-30)}$$

$$(\forall y\ (interior\_point(y) \equiv\ (\exists l (point(y) \wedge line(l) \wedge in(y,l) \wedge (\forall z (line(z) \wedge in(y,z) \supset\ z = l)))))). \quad \text{(CPSN-31)}$$

$$(\forall p\ (intersecting\_point(p) \equiv point(p) \wedge (\exists l_1 \exists l_2 (line(l_1) \wedge line(l_2) \wedge l_1 \neq l_2 \wedge in(p,l_1) \wedge in(p,l_2))))). \quad \text{(CPSN-32)}$$

$$(\forall l_1 \forall l_2\ (overlap(l_1,l_2) \equiv intersect(l_1,l_2) \wedge (\exists p (point(p) \wedge in(p,l_1) \wedge \neg in(p,l_2))))). \quad \text{(CPSN-33)}$$

$$(\forall x\ (pendant\_point(x) \equiv point(x) \wedge (\forall y \forall z (collinear(x,y) \wedge collinear(x,z) \supset\ y = z)))). \quad \text{(CPSN-34)}$$

For reference, the remainder of the axioms for this theory can be found in Figure 13.6.

C.2 Extensions of MoSt

The following are possible extensions of MoSt.

C.2.1 K_4-free Graphs in MoSt

This is an extension of MoSt that is K_4-free.

$$\begin{aligned}\neg(\exists a_1\exists a_2\exists a_3\exists a_4\exists b_1\exists b_2\exists b_3\exists b_4\exists b_5\exists b_6\ & atom(a_1)\wedge atom(a_2)\wedge atom(a_3)\wedge\\ & atom(a_4)\wedge(a_1\neq a_2)\wedge(a_1\neq a_3)\wedge(a_1\neq a_4)\wedge\\ & (a_2\neq a_3)\wedge(a_2\neq a_4)\wedge(a_3\neq a_4)\wedge bond(b_1)\wedge\\ & bond(b_2)\wedge bond(b_3)\wedge bond(b_4)\wedge bond(b_5)\wedge\\ & bond(b_6)\wedge mol(a_1,b_1)\wedge mol(a_2,b_1)\wedge mol(a_2,b_2)\wedge\\ & mol(a_3,b_2)\wedge mol(a_3,b_3)\wedge mol(a_1,b_3)\wedge mol(a_4,b_4)\wedge\\ & mol(a_1,b_4)\wedge mol(a_4,b_5)\wedge mol(a_2,b_5)\wedge mol(a_4,b_6)\wedge\\ & mol(a_3,b_6))\end{aligned} \tag{K4-free}$$

C.2.2 $T_{most\_polycyclic}$

All functional groups are rings. $T_{most\_polycyclic} = T_{most} \cup \forall x\ group(x) \supset ring(x)$

C.2.3 T_{most_chains}

All functional groups are chains. $T_{most\_polycyclic} = T_{most} \cup \forall x\ group(x) \supset chain(x)$

Appendix D

Proofs

D.1 Proof for Theorem 6.4.1

Recall Theorem 6.4.1:

$$\forall x \forall g\, (fork(x) \wedge mol(x,g) \wedge group(g)) \supset \exists b \exists y\, atom(y) \wedge bond(b) \wedge mol(x,b) \wedge mol(y,b) \wedge \neg mol(y,g). \quad \text{(M-ThF)}$$

The following proof, as well as the input file, is available online in COLORE at `http://colore.oor.net/most/interprets/fork_theorem`:

Proof D.1.1: $T_{most\_group} \models$ Theorem 6.4.1

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 15648 was started by cchui on MacBook-Pro.local,
Fri Nov 30 12:16:04 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
    ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 2.85 (+ 0.03) seconds.
% Length of proof is 41.
% Level of proof is 5.
% Maximum clause weight is 64.
% Given clauses 63.

4 (all a1 all a2 all a3 all a4 all g all b1 all b2 all b3 (group(g) & atom(
    ↪ a1) & atom(a2) & atom(a3) & atom(a4) & mol(a1,g) & mol(a2,g) & mol(a3
    ↪ ,g) & mol(a4,g) & bond(b1) & mol(b1,g) & mol(a1,b1) & mol(a2,b1) &
    ↪ bond(b2) & mol(b2,g) & mol(a1,b2) & mol(a3,b2) & bond(b3) & mol(b3,g)
    ↪  & mol(a1,b3) & mol(a4,b3) -> a4 = a3 | a2 = a3 | a4 = a2)) # label(
    ↪ non_clause). [assumption].
```

```
5 (all x all y all b all g (atom(x) & atom(y) & bond(b) & group(g) & x != y
   ↪  & mol(x,g) & mol(y,g) & mol(x,b) & mol(y,b) -> mol(b,g))) # label(
   ↪ non_clause). [assumption].
6 (all x (fork(x) -> atom(x) & (exists b1 exists b2 exists b3 exists a1
   ↪ exists a2 exists a3 (atom(a1) & atom(a2) & atom(a3) & bond(b1) & bond
   ↪ (b2) & bond(b3) & a1 != a2 & a2 != a3 & a1 != a3 & a1 != x & a2 != x
   ↪ & a3 != x & mol(x,b1) & mol(a1,b1) & mol(x,b2) & mol(a2,b2) & mol(x,
   ↪ b3) & mol(a3,b3))))) # label(non_clause). [assumption].
11 -(all x all g (fork(x) & mol(x,g) & group(g) -> (exists b exists y (atom
   ↪ (y) & bond(b) & mol(x,b) & mol(y,b) & -mol(y,g))))) # label(
   ↪ non_clause). [assumption].
12 fork(c1). [clausify(11)].
13 -fork(x) | atom(x). [clausify(6)].
15 -fork(x) | atom(f5(x)). [clausify(6)].
16 -fork(x) | atom(f6(x)). [clausify(6)].
18 -fork(x) | bond(f2(x)). [clausify(6)].
19 -fork(x) | bond(f3(x)). [clausify(6)].
21 -fork(x) | f6(x) != f5(x). [clausify(6)].
24 -fork(x) | f5(x) != x. [clausify(6)].
25 -fork(x) | f6(x) != x. [clausify(6)].
28 -fork(x) | mol(x,f2(x)). [clausify(6)].
29 -fork(x) | mol(f5(x),f2(x)). [clausify(6)].
30 -fork(x) | mol(x,f3(x)). [clausify(6)].
31 -fork(x) | mol(f6(x),f3(x)). [clausify(6)].
36 -group(x) | -atom(y) | -atom(z) | -atom(u) | -atom(w) | -mol(y,x) | -mol
   ↪ (z,x) | -mol(u,x) | -mol(w,x) | -bond(v5) | -mol(v5,x) | -mol(y,v5) |
   ↪  -mol(z,v5) | -bond(v6) | -mol(v6,x) | -mol(y,v6) | -mol(u,v6) | -
   ↪ bond(v7) | -mol(v7,x) | -mol(y,v7) | -mol(w,v7) | w = u | u = z | w =
   ↪  z. [clausify(4)].
37 -atom(x) | -atom(y) | -bond(z) | -group(u) | y = x | -mol(x,u) | -mol(y,
   ↪ u) | -mol(x,z) | -mol(y,z) | mol(z,u). [clausify(5)].
42 mol(c1,c2). [clausify(11)].
43 group(c2). [clausify(11)].
44 -atom(x) | -bond(y) | -mol(c1,y) | -mol(x,y) | mol(x,c2). [clausify(11)
   ↪ ].
45 atom(c1). [resolve(12,a,13,a)].
47 atom(f5(c1)). [resolve(12,a,15,a)].
48 atom(f6(c1)). [resolve(12,a,16,a)].
50 bond(f2(c1)). [resolve(12,a,18,a)].
51 bond(f3(c1)). [resolve(12,a,19,a)].
53 f6(c1) != f5(c1). [resolve(12,a,21,a)].
56 f5(c1) != c1. [resolve(12,a,24,a)].
57 f6(c1) != c1. [resolve(12,a,25,a)].
60 mol(c1,f2(c1)). [resolve(12,a,28,a)].
61 mol(f5(c1),f2(c1)). [resolve(12,a,29,a)].
62 mol(c1,f3(c1)). [resolve(12,a,30,a)].
63 mol(f6(c1),f3(c1)). [resolve(12,a,31,a)].
79 -atom(x) | -bond(y) | c1 = x | -mol(x,c2) | -mol(x,y) | -mol(c1,y) | mol
   ↪ (y,c2). [resolve(42,a,37,g),unit_del(b,45),unit_del(d,43)].
87 -atom(x) | -atom(y) | -atom(z) | -mol(x,c2) | -mol(y,c2) | -mol(z,c2) |
   ↪ -bond(u) | -mol(u,c2) | -mol(x,u) | -mol(y,u) | -bond(w) | -mol(w,c2)
   ↪  | -mol(x,w) | -mol(z,w) | -bond(v5) | -mol(v5,c2) | -mol(x,v5) | -
   ↪ mol(c1,v5) | c1 = z | z = y | c1 = y. [resolve(42,a,36,i),unit_del(a
   ↪ ,43),unit_del(e,45)].
```

```
351 mol(f5(c1),c2). [resolve(61,a,44,d),unit_del(a,47),unit_del(b,50),
    ↪ unit_del(c,60)].
411 mol(f6(c1),c2). [resolve(63,a,44,d),unit_del(a,48),unit_del(b,51),
    ↪ unit_del(c,62)].
473 mol(f3(c1),c2). [resolve(79,e,63,a),flip(c),unit_del(a,48),unit_del(b
    ↪ ,51),unit_del(c,57),unit_del(d,411),unit_del(e,62)].
474 mol(f2(c1),c2). [resolve(79,e,61,a),flip(c),unit_del(a,47),unit_del(b
    ↪ ,50),unit_del(c,56),unit_del(d,351),unit_del(e,60)].
632 $F. [ur(87,a,45,a,b,47,a,c,48,a,d,42,a,e,351,a,f,411,a,g,50,a,i,60,a,j
    ↪ ,61,a,k,51,a,l,473,a,m,62,a,n,63,a,o,51,a,p,473,a,q,62,a,r,62,a,s,57,
    ↪ a(flip),t,53,a,u,56,a(flip)),unit_del(a,474)].

============================== end of proof ==========================
```

D.2 Proofs for T_{most_graph} Verification

Let Δ_1 be the set of translation definitions:

$$(\forall x(point(x) \equiv atom(x))).$$
$$(\forall x(line(x) \equiv bond(x))).$$
$$(\forall x \forall y\ (in(x,y) \equiv mol(x,y))).$$

$$T_{most\_graph} \cup \Delta_1 \models T_{nonisolated\_loopless}$$

The following proofs, as well as the input files, are available online in COLORE at `http://colore.oor.net/molecular_graph/interprets`:

- $T_{most\_graph} \cup \Delta_1 \models$ Axiom #1 in $T_{nonisolated\_loopless}$
- $T_{most\_graph} \cup \Delta_1 \models$ Axiom #2 in $T_{nonisolated\_loopless}$
- $T_{most\_graph} \cup \Delta_1 \models$ Axiom #3 in $T_{nonisolated\_loopless}$
- $T_{most\_graph} \cup \Delta_1 \models$ Axiom #4 in $T_{nonisolated\_loopless}$
- $T_{most\_graph} \cup \Delta_1 \models$ Axiom #5 in $T_{nonisolated\_loopless}$
- $T_{most\_graph} \cup \Delta_1 \models$ Axiom #6 in $T_{nonisolated\_loopless}$
- $T_{most\_graph} \cup \Delta_1 \models$ Axiom #7 in $T_{nonisolated\_loopless}$
- $T_{most\_graph} \cup \Delta_1 \models$ Axiom #8 in $T_{nonisolated\_loopless}$
- $T_{most\_graph} \cup \Delta_1 \models$ Axiom #9 in $T_{nonisolated\_loopless}$

Proof D.2.1: $T_{most\_graph} \cup \Delta_1 \models$ Axiom #1 in $T_{nonisolated\_loopless}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 17218 was started by cchui on MacBook-Pro.local,
Fri Oct 19 22:45:54 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.00 (+ 0.00) seconds.
% Length of proof is 18.
% Level of proof is 5.
% Maximum clause weight is 6.
% Given clauses 30.

9 (all b (bond(b) -> (exists x exists y (atom(x) & atom(y) & mol(x,b) & mol
   ↪ (y,b) & y != x)))) # label(non_clause). [assumption].
```

```
10 (all x (point(x) <-> atom(x))) # label(non_clause). [assumption].
11 (all x (line(x) <-> bond(x))) # label(non_clause). [assumption].
12 (all x all y (in(x,y) <-> mol(x,y))) # label(non_clause). [assumption].
13 (all l (line(l) -> (exists x (point(x) & in(x,l))))) # label(non_clause)
   ↪ # label(goal). [goal].
14 point(x) | -atom(x). [clausify(10)].
16 -point(x) | -in(x,c1). [deny(13)].
18 -line(x) | bond(x). [clausify(11)].
19 line(c1). [deny(13)].
36 -bond(x) | atom(f8(x)). [clausify(9)].
38 -bond(x) | mol(f8(x),x). [clausify(9)].
41 in(x,y) | -mol(x,y). [clausify(12)].
42 -in(x,c1) | -atom(x). [resolve(16,a,14,a)].
43 bond(c1). [resolve(19,a,18,a)].
48 mol(f8(c1),c1). [resolve(43,a,38,a)].
50 atom(f8(c1)). [resolve(43,a,36,a)].
62 -in(f8(c1),c1). [ur(42,b,50,a)].
76 $F. [resolve(48,a,41,b),unit_del(a,62)].

============================== end of proof ==========================
```

Proof D.2.2: $T_{most\_graph} \cup \Delta_1 \models$ Axiom #2 in $T_{nonisolated\_loopless}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 17223 was started by cchui on MacBook-Pro.local,
Fri Oct 19 22:46:15 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.01 (+ 0.00) seconds.
% Length of proof is 27.
% Level of proof is 3.
% Maximum clause weight is 26.
% Given clauses 27.

5 (all x all y all z all b (atom(x) & atom(y) & atom(z) & bond(b) & mol(x,b
   ↪ ) & mol(y,b) & mol(z,b) -> x = y | x = z | y = z)) # label(non_clause
   ↪ ). [assumption].
10 (all x (point(x) <-> atom(x))) # label(non_clause). [assumption].
11 (all x (line(x) <-> bond(x))) # label(non_clause). [assumption].
12 (all x all y (in(x,y) <-> mol(x,y))) # label(non_clause). [assumption].
13 (all x all y all z all l (point(x) & point(y) & point(z) & line(l) & in(
   ↪ x,l) & in(y,l) & in(z,l) -> z = x | z = y | x = y)) # label(
   ↪ non_clause) # label(goal). [goal].
15 -point(x) | atom(x). [clausify(10)].
16 point(c1). [deny(13)].
17 point(c2). [deny(13)].
```

```
18 point(c3). [deny(13)].
20 -line(x) | bond(x). [clausify(11)].
21 line(c4). [deny(13)].
23 -in(x,y) | mol(x,y). [clausify(12)].
24 in(c1,c4). [deny(13)].
25 in(c2,c4). [deny(13)].
26 in(c3,c4). [deny(13)].
31 -atom(x) | -atom(y) | -atom(z) | -bond(u) | -mol(x,u) | -mol(y,u) | -mol
   ↪ (z,u) | y = x | z = x | z = y. [clausify(5)].
47 c3 != c1. [deny(13)].
48 c3 != c2. [deny(13)].
49 c2 != c1. [deny(13)].
50 atom(c1). [resolve(16,a,15,a)].
51 atom(c2). [resolve(17,a,15,a)].
52 atom(c3). [resolve(18,a,15,a)].
53 bond(c4). [resolve(21,a,20,a)].
54 mol(c1,c4). [resolve(24,a,23,a)].
55 mol(c2,c4). [resolve(25,a,23,a)].
56 mol(c3,c4). [resolve(26,a,23,a)].
94 $F. [ur(31,a,52,a,b,50,a,c,51,a,d,53,a,f,54,a,g,55,a,h,47,a(flip),i,48,a
   ↪ (flip),j,49,a),unit_del(a,56)].

============================== end of proof ==========================
```

Proof D.2.3: $T_{most\_graph} \cup \Delta_1 \models$ Axiom #3 in $T_{nonisolated\_loopless}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 17228 was started by cchui on MacBook-Pro.local,
Fri Oct 19 22:46:30 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.03 (+ 0.00) seconds.
% Length of proof is 26.
% Level of proof is 5.
% Maximum clause weight is 13.
% Given clauses 56.

9 (all b (bond(b) -> (exists x exists y (atom(x) & atom(y) & mol(x,b) & mol
   ↪ (y,b) & y != x)))) # label(non_clause). [assumption].
10 (all x (point(x) <-> atom(x))) # label(non_clause). [assumption].
11 (all x (line(x) <-> bond(x))) # label(non_clause). [assumption].
12 (all x all y (in(x,y) <-> mol(x,y))) # label(non_clause). [assumption].
13 (all l (line(l) -> (exists x exists y (point(x) & point(y) & x != y & in
   ↪ (x,l) & in(y,l))))) # label(non_clause) # label(goal). [goal].
15 -line(x) | bond(x). [clausify(11)].
16 line(c1). [deny(13)].
```

```
32 -bond(x) | atom(f7(x)). [clausify(9)].
33 -bond(x) | atom(f8(x)). [clausify(9)].
34 -bond(x) | mol(f7(x),x). [clausify(9)].
35 -bond(x) | mol(f8(x),x). [clausify(9)].
36 -bond(x) | f8(x) != f7(x). [clausify(9)].
38 point(x) | -atom(x). [clausify(10)].
40 in(x,y) | -mol(x,y). [clausify(12)].
41 -point(x) | -point(y) | y = x | -in(x,c1) | -in(y,c1). [deny(13)].
42 bond(c1). [resolve(16,a,15,a)].
46 f8(c1) != f7(c1). [resolve(42,a,36,a)].
47 mol(f8(c1),c1). [resolve(42,a,35,a)].
48 mol(f7(c1),c1). [resolve(42,a,34,a)].
49 atom(f8(c1)). [resolve(42,a,33,a)].
50 atom(f7(c1)). [resolve(42,a,32,a)].
57 point(f8(c1)). [resolve(49,a,38,b)].
62 point(f7(c1)). [resolve(50,a,38,b)].
75 in(f8(c1),c1). [resolve(47,a,40,b)].
88 in(f7(c1),c1). [resolve(48,a,40,b)].
153 $F. [ur(41,a,62,a,b,57,a,c,46,a,e,75,a),unit_del(a,88)].

============================== end of proof ==========================
```

Proof D.2.4: $T_{most\_graph} \cup \Delta_1 \models$ Axiom #4 in $T_{nonisolated\_loopless}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 17233 was started by cchui on MacBook-Pro.local,
Fri Oct 19 22:46:46 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.00 (+ 0.00) seconds.
% Length of proof is 11.
% Level of proof is 3.
% Maximum clause weight is 6.
% Given clauses 25.

2 (all x all y (mol(x,y) -> mol(y,x))) # label(non_clause). [assumption].
12 (all x all y (in(x,y) <-> mol(x,y))) # label(non_clause). [assumption].
13 (all x all y (in(x,y) -> in(y,x))) # label(non_clause) # label(goal). [
   ↪ goal].
19 -mol(x,y) | mol(y,x). [clausify(2)].
38 -in(x,y) | mol(x,y). [clausify(12)].
39 in(x,y) | -mol(x,y). [clausify(12)].
40 in(c1,c2). [deny(13)].
41 -in(c2,c1). [deny(13)].
45 mol(c1,c2). [resolve(40,a,38,a)].
46 -mol(c2,c1). [ur(39,a,41,a)].
```

```
48 $F. [ur(19,b,46,a),unit_del(a,45)].

============================== end of proof ==========================
```

Proof D.2.5: $T_{most\_graph} \cup \Delta_1 \models$ Axiom #5 in $T_{nonisolated\_loopless}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 17238 was started by cchui on MacBook-Pro.local,
Fri Oct 19 22:47:01 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.00 (+ 0.00) seconds.
% Length of proof is 8.
% Level of proof is 3.
% Maximum clause weight is 6.
% Given clauses 20.

1 (all x mol(x,x)) # label(non_clause). [assumption].
12 (all x all y (in(x,y) <-> mol(x,y))) # label(non_clause). [assumption].
13 (all x (point(x) | line(x) -> in(x,x))) # label(non_clause) # label(goal
   ↪ ). [goal].
20 mol(x,x). [clausify(1)].
41 in(x,y) | -mol(x,y). [clausify(12)].
42 -in(c1,c1). [deny(13)].
46 in(x,x). [resolve(41,b,20,a)].
47 $F. [resolve(46,a,42,a)].

============================== end of proof ==========================
```

Proof D.2.6: $T_{most\_graph} \cup \Delta_1 \models$ Axiom #6 in $T_{nonisolated\_loopless}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 17244 was started by cchui on MacBook-Pro.local,
Fri Oct 19 22:47:16 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.02 (+ 0.00) seconds.
% Length of proof is 25.
```

```
% Level of proof is 5.
% Maximum clause weight is 10.
% Given clauses 35.

3 (all x all y (mol(x,y) & atom(x) & atom(y) -> x = y)) # label(non_clause)
   ↪ . [assumption].
9 (all b (bond(b) -> (exists x exists y (atom(x) & atom(y) & mol(x,b) & mol
   ↪ (y,b) & y != x)))) # label(non_clause). [assumption].
10 (all x (point(x) <-> atom(x))) # label(non_clause). [assumption].
11 (all x (line(x) <-> bond(x))) # label(non_clause). [assumption].
13 (all p (point(p) -> -line(p))) # label(non_clause) # label(goal). [goal
   ↪ ].
15 -point(x) | atom(x). [clausify(10)].
16 point(c1). [deny(13)].
18 -line(x) | bond(x). [clausify(11)].
19 line(c1). [deny(13)].
24 -mol(x,y) | -atom(x) | -atom(y) | y = x. [clausify(3)].
37 -bond(x) | atom(f7(x)). [clausify(9)].
38 -bond(x) | atom(f8(x)). [clausify(9)].
39 -bond(x) | mol(f7(x),x). [clausify(9)].
40 -bond(x) | mol(f8(x),x). [clausify(9)].
41 -bond(x) | f8(x) != f7(x). [clausify(9)].
42 atom(c1). [resolve(16,a,15,a)].
43 bond(c1). [resolve(19,a,18,a)].
50 f8(c1) != f7(c1). [resolve(43,a,41,a)].
51 mol(f8(c1),c1). [resolve(43,a,40,a)].
52 mol(f7(c1),c1). [resolve(43,a,39,a)].
53 atom(f8(c1)). [resolve(43,a,38,a)].
54 atom(f7(c1)). [resolve(43,a,37,a)].
82 -mol(f7(c1),f8(c1)). [ur(24,b,54,a,c,53,a,d,50,a)].
107 f8(c1) = c1. [resolve(51,a,24,a),flip(c),unit_del(a,53),unit_del(b,42)
   ↪ ].
108 $F. [back_rewrite(82),rewrite([107(4)]),unit_del(a,52)].

============================== end of proof ==========================
```

Proof D.2.7: $T_{most\_graph} \cup \Delta_1 \models$ Axiom #7 in $T_{nonisolated\_loopless}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 17249 was started by cchui on MacBook-Pro.local,
Fri Oct 19 22:47:29 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.00 (+ 0.00) seconds.
% Length of proof is 15.
% Level of proof is 3.
```

```
% Maximum clause weight is 10.
% Given clauses 21.

3 (all x all y (mol(x,y) & atom(x) & atom(y) -> x = y)) # label(non_clause)
   ↪ . [assumption].
10 (all x (point(x) <-> atom(x))) # label(non_clause). [assumption].
12 (all x all y (in(x,y) <-> mol(x,y))) # label(non_clause). [assumption].
13 (all x all y (in(x,y) & point(x) & point(y) -> x = y)) # label(
   ↪ non_clause) # label(goal). [goal].
15 -point(x) | atom(x). [clausify(10)].
16 point(c1). [deny(13)].
17 point(c2). [deny(13)].
21 -in(x,y) | mol(x,y). [clausify(12)].
22 in(c1,c2). [deny(13)].
25 -mol(x,y) | -atom(x) | -atom(y) | y = x. [clausify(3)].
43 c2 != c1. [deny(13)].
44 atom(c1). [resolve(16,a,15,a)].
45 atom(c2). [resolve(17,a,15,a)].
46 mol(c1,c2). [resolve(22,a,21,a)].
57 $F. [ur(25,b,44,a,c,45,a,d,43,a),unit_del(a,46)].

============================== end of proof ==========================
```

Proof D.2.8: $T_{most\_graph} \cup \Delta_1 \models$ Axiom #8 in $T_{nonisolated\_loopless}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 17254 was started by cchui on MacBook-Pro.local,
Fri Oct 19 22:47:43 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.00 (+ 0.00) seconds.
% Length of proof is 15.
% Level of proof is 3.
% Maximum clause weight is 10.
% Given clauses 21.

4 (all x all y (mol(x,y) & bond(x) & bond(y) -> x = y)) # label(non_clause)
   ↪ . [assumption].
11 (all x (line(x) <-> bond(x))) # label(non_clause). [assumption].
12 (all x all y (in(x,y) <-> mol(x,y))) # label(non_clause). [assumption].
13 (all x all y (in(x,y) & line(x) & line(y) -> x = y)) # label(non_clause)
   ↪  # label(goal). [goal].
17 -line(x) | bond(x). [clausify(11)].
18 line(c1). [deny(13)].
19 line(c2). [deny(13)].
21 -in(x,y) | mol(x,y). [clausify(12)].
```

```
22 in(c1,c2). [deny(13)].
26 -mol(x,y) | -bond(x) | -bond(y) | y = x. [clausify(4)].
43 c2 != c1. [deny(13)].
44 bond(c1). [resolve(18,a,17,a)].
45 bond(c2). [resolve(19,a,17,a)].
46 mol(c1,c2). [resolve(22,a,21,a)].
67 $F. [ur(26,b,44,a,c,45,a,d,43,a),unit_del(a,46)].

============================== end of proof ==========================
```

Proof D.2.9: $T_{most\_graph} \cup \Delta_1 \models$ Axiom #9 in $T_{nonisolated\_loopless}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 17262 was started by cchui on MacBook-Pro.local,
Fri Oct 19 22:47:55 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
    ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.00 (+ 0.00) seconds.
% Length of proof is 18.
% Level of proof is 5.
% Maximum clause weight is 6.
% Given clauses 28.

6 (all x (atom(x) -> (exists b exists y (atom(y) & bond(b) & mol(x,b) & mol
    ↪ (y,b))))) # label(non_clause). [assumption].
10 (all x (point(x) <-> atom(x))) # label(non_clause). [assumption].
11 (all x (line(x) <-> bond(x))) # label(non_clause). [assumption].
12 (all x all y (in(x,y) <-> mol(x,y))) # label(non_clause). [assumption].
13 (all p (point(p) -> (exists l (line(l) & in(p,l))))) # label(non_clause)
    ↪  # label(goal). [goal].
15 -point(x) | atom(x). [clausify(10)].
16 point(c1). [deny(13)].
17 line(x) | -bond(x). [clausify(11)].
19 -line(x) | -in(c1,x). [deny(13)].
26 -atom(x) | bond(f1(x)). [clausify(6)].
27 -atom(x) | mol(x,f1(x)). [clausify(6)].
41 in(x,y) | -mol(x,y). [clausify(12)].
42 atom(c1). [resolve(16,a,15,a)].
43 -in(c1,x) | -bond(x). [resolve(19,a,17,a)].
48 mol(c1,f1(c1)). [resolve(42,a,27,a)].
49 bond(f1(c1)). [resolve(42,a,26,a)].
62 -in(c1,f1(c1)). [ur(43,b,49,a)].
68 $F. [resolve(48,a,41,b),unit_del(a,62)].

============================== end of proof ==========================
```

D.3 Proofs for T_{most_graph} Reduction

Let Π_1 be the set of translation definitions:

$$(\forall x(atom(x) \equiv point(x))).$$
$$(\forall x(bond(x) \equiv line(x))).$$
$$(\forall x \forall y\ (mol(x,y) \equiv in(x,y)))$$

$$T_{nonisolated\_loopless} \cup \Pi_1 \models T_{most\_graph}$$

The following proofs, as well as the input files, are available online in COLORE at `http://colore.oor.net/molecular_graph/interprets`:

- $T_{nonisolated\_loopless} \cup \Pi_1 \models$ Axiom #1 in T_{most_graph}
- $T_{nonisolated\_loopless} \cup \Pi_1 \models$ Axiom #2 in T_{most_graph}
- $T_{nonisolated\_loopless} \cup \Pi_1 \models$ Axiom #3 in T_{most_graph}
- $T_{nonisolated\_loopless} \cup \Pi_1 \models$ Axiom #4 in T_{most_graph}
- $T_{nonisolated\_loopless} \cup \Pi_1 \models$ Axiom #5 in T_{most_graph}
- $T_{nonisolated\_loopless} \cup \Pi_1 \models$ Axiom #6 in T_{most_graph}
- $T_{nonisolated\_loopless} \cup \Pi_1 \models$ Axiom #7 in T_{most_graph}
- $T_{nonisolated\_loopless} \cup \Pi_1 \models$ Axiom #8 in T_{most_graph}
- $T_{nonisolated\_loopless} \cup \Pi_1 \models$ Axiom #9 in T_{most_graph}

Proof D.3.1: $T_{nonisolated\_loopless} \cup \Pi_1 \models$ Axiom #1 in T_{most_graph}

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 17282 was started by cchui on MacBook-Pro.local,
Fri Oct 19 22:50:15 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.00 (+ 0.00) seconds.
% Length of proof is 6.
% Level of proof is 3.
% Maximum clause weight is 6.
% Given clauses 20.

12 (all x all y (mol(x,y) <-> in(x,y) | in(y,x) | x = y)) # label(
   ↪ non_clause). [assumption].
```

```
14 (all x mol(x,x)) # label(non_clause) # label(goal). [goal].
38 mol(x,y) | y != x. [clausify(12)].
42 -mol(c3,c3). [deny(14)].
43 mol(x,x). [xx_res(38,b)].
44 $F. [resolve(43,a,42,a)].

============================== end of proof ==========================
```

Proof D.3.2: $T_{nonisolated\_loopless} \cup \Pi_1 \models$ Axiom #2 in T_{most_graph}

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 17290 was started by cchui on MacBook-Pro.local,
Fri Oct 19 22:50:40 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.00 (+ 0.00) seconds.
% Length of proof is 11.
% Level of proof is 3.
% Maximum clause weight is 6.
% Given clauses 27.

4 (all x all y (in(x,y) -> in(y,x))) # label(non_clause). [assumption].
12 (all x all y (mol(x,y) <-> in(x,y))) # label(non_clause). [assumption].
14 (all x all y (mol(x,y) -> mol(y,x))) # label(non_clause) # label(goal).
   ↪ [goal].
27 -in(x,y) | in(y,x). [clausify(4)].
35 -mol(x,y) | in(x,y). [clausify(12)].
36 mol(x,y) | -in(x,y). [clausify(12)].
40 mol(c3,c4). [deny(14)].
41 -mol(c4,c3). [deny(14)].
58 in(c3,c4). [resolve(40,a,35,a)].
59 -in(c4,c3). [ur(36,a,41,a)].
64 $F. [ur(27,b,59,a),unit_del(a,58)].

============================== end of proof ==========================
```

Proof D.3.3: $T_{nonisolated\_loopless} \cup \Pi_1 \models$ Axiom #3 in T_{most_graph}

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 17295 was started by cchui on MacBook-Pro.local,
Fri Oct 19 22:50:56 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================
```

```
============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.00 (+ 0.00) seconds.
% Length of proof is 15.
% Level of proof is 3.
% Maximum clause weight is 10.
% Given clauses 22.

7 (all x all y (in(x,y) & point(x) & point(y) -> x = y)) # label(non_clause
   ↪ ). [assumption].
10 (all x (atom(x) <-> point(x))) # label(non_clause). [assumption].
12 (all x all y (mol(x,y) <-> in(x,y))) # label(non_clause). [assumption].
14 (all x all y (mol(x,y) & atom(x) & atom(y) -> x = y)) # label(non_clause
   ↪ ) # label(goal). [goal].
16 -atom(x) | point(x). [clausify(10)].
17 atom(c3). [deny(14)].
18 atom(c4). [deny(14)].
22 -mol(x,y) | in(x,y). [clausify(12)].
23 mol(c3,c4). [deny(14)].
36 -in(x,y) | -point(x) | -point(y) | y = x. [clausify(7)].
43 c4 != c3. [deny(14)].
44 point(c3). [resolve(17,a,16,a)].
45 point(c4). [resolve(18,a,16,a)].
46 in(c3,c4). [resolve(23,a,22,a)].
69 $F. [ur(36,b,44,a,c,45,a,d,43,a),unit_del(a,46)].

============================== end of proof ==========================
```

Proof D.3.4: $T_{nonisolated\_loopless} \cup \Pi_1 \models$ Axiom #4 in T_{most_graph}

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 17300 was started by cchui on MacBook-Pro.local,
Fri Oct 19 22:51:09 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.00 (+ 0.00) seconds.
% Length of proof is 15.
% Level of proof is 3.
% Maximum clause weight is 10.
% Given clauses 22.

8 (all x all y (in(x,y) & line(x) & line(y) -> x = y)) # label(non_clause).
   ↪  [assumption].
```

```
11 (all x (bond(x) <-> line(x))) # label(non_clause). [assumption].
12 (all x all y (mol(x,y) <-> in(x,y))) # label(non_clause). [assumption].
14 (all x all y (mol(x,y) & bond(x) & bond(y) -> x = y)) # label(non_clause
   ↪ ) # label(goal). [goal].
18 -bond(x) | line(x). [clausify(11)].
19 bond(c3). [deny(14)].
20 bond(c4). [deny(14)].
22 -mol(x,y) | in(x,y). [clausify(12)].
23 mol(c3,c4). [deny(14)].
37 -in(x,y) | -line(x) | -line(y) | y = x. [clausify(8)].
43 c4 != c3. [deny(14)].
44 line(c3). [resolve(19,a,18,a)].
45 line(c4). [resolve(20,a,18,a)].
46 in(c3,c4). [resolve(23,a,22,a)].
79 $F. [ur(37,b,44,a,c,45,a,d,43,a),unit_del(a,46)].

============================== end of proof ==========================
```

Proof D.3.5: $T_{nonisolated\_loopless} \cup \Pi_1 \models$ Axiom #5 in T_{most_graph}

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 17307 was started by cchui on MacBook-Pro.local,
Fri Oct 19 22:51:23 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.01 (+ 0.00) seconds.
% Length of proof is 27.
% Level of proof is 3.
% Maximum clause weight is 26.
% Given clauses 28.

2 (all x all y all z all l (point(x) & point(y) & point(z) & line(l) & in(x
   ↪ ,l) & in(y,l) & in(z,l) -> z = x | z = y | x = y)) # label(non_clause
   ↪ ). [assumption].
10 (all x (atom(x) <-> point(x))) # label(non_clause). [assumption].
11 (all x (bond(x) <-> line(x))) # label(non_clause). [assumption].
12 (all x all y (mol(x,y) <-> in(x,y))) # label(non_clause). [assumption].
14 (all x all y all z all b (atom(x) & atom(y) & atom(z) & bond(b) & mol(x,
   ↪ b) & mol(y,b) & mol(z,b) -> x = y | x = z | y = z)) # label(
   ↪ non_clause) # label(goal). [goal].
16 -atom(x) | point(x). [clausify(10)].
17 atom(c3). [deny(14)].
18 atom(c4). [deny(14)].
19 atom(c5). [deny(14)].
21 -bond(x) | line(x). [clausify(11)].
22 bond(c6). [deny(14)].
```

```
24 -mol(x,y) | in(x,y). [clausify(12)].
25 mol(c3,c6). [deny(14)].
26 mol(c4,c6). [deny(14)].
27 mol(c5,c6). [deny(14)].
30 -point(x) | -point(y) | -point(z) | -line(u) | -in(x,u) | -in(y,u) | -in
   ↪ (z,u) | z = x | z = y | y = x. [clausify(2)].
47 c4 != c3. [deny(14)].
48 c5 != c3. [deny(14)].
49 c5 != c4. [deny(14)].
50 point(c3). [resolve(17,a,16,a)].
51 point(c4). [resolve(18,a,16,a)].
52 point(c5). [resolve(19,a,16,a)].
53 line(c6). [resolve(22,a,21,a)].
54 in(c3,c6). [resolve(25,a,24,a)].
55 in(c4,c6). [resolve(26,a,24,a)].
56 in(c5,c6). [resolve(27,a,24,a)].
103 $F. [ur(30,a,52,a,b,50,a,c,51,a,d,53,a,f,54,a,g,55,a,h,49,a(flip),i,47,
   ↪ a,j,48,a(flip)),unit_del(a,56)].

============================== end of proof ==========================
```

Proof D.3.6: $T_{nonisolated\_loopless} \cup \Pi_1 \models$ Axiom #6 in T_{most_graph}

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 17312 was started by cchui on MacBook-Pro.local,
Fri Oct 19 22:51:38 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.01 (+ 0.00) seconds.
% Length of proof is 20.
% Level of proof is 6.
% Maximum clause weight is 6.
% Given clauses 54.

9 (all p (point(p) -> (exists l (line(l) & in(p,l))))) # label(non_clause).
   ↪  [assumption].
10 (all x (atom(x) <-> point(x))) # label(non_clause). [assumption].
11 (all x (bond(x) <-> line(x))) # label(non_clause). [assumption].
12 (all x all y (mol(x,y) <-> in(x,y))) # label(non_clause). [assumption].
14 (all x (atom(x) -> (exists b exists y (atom(y) & bond(b) & mol(x,b) &
   ↪ mol(y,b))))) # label(non_clause) # label(goal). [goal].
16 -atom(x) | point(x). [clausify(10)].
17 atom(c3). [deny(14)].
18 -atom(x) | -bond(y) | -mol(c3,y) | -mol(x,y). [deny(14)].
19 bond(x) | -line(x). [clausify(11)].
22 -bond(x) | -mol(c3,x) | -mol(c3,x). [resolve(18,a,17,a)].
```

```
37 -point(x) | line(f4(x)). [clausify(9)].
38 -point(x) | in(x,f4(x)). [clausify(9)].
40 mol(x,y) | -in(x,y). [clausify(12)].
44 point(c3). [resolve(17,a,16,a)].
46 -mol(c3,x) | -mol(c3,x) | -line(x). [resolve(22,a,19,a)].
47 -mol(c3,x) | -line(x). [copy(46),merge(b)].
64 in(c3,f4(c3)). [resolve(44,a,38,a)].
65 line(f4(c3)). [resolve(44,a,37,a)].
117 -mol(c3,f4(c3)). [ur(47,b,65,a)].
122 $F. [ur(40,a,117,a),unit_del(a,64)].

============================== end of proof ==========================
```

Proof D.3.7: $T_{nonisolated\_loopless} \cup \Pi_1 \models$ Axiom #7 in T_{most_graph}

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 17318 was started by cchui on MacBook-Pro.local,
Fri Oct 19 22:51:53 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
    ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.02 (+ 0.00) seconds.
% Length of proof is 19.
% Level of proof is 6.
% Maximum clause weight is 10.
% Given clauses 75.

3 (all l (line(l) -> (exists x exists y (point(x) & point(y) & x != y & in(
    ↪ x,l) & in(y,l))))) # label(non_clause). [assumption].
10 (all x (atom(x) <-> point(x))) # label(non_clause). [assumption].
11 (all x (bond(x) <-> line(x))) # label(non_clause). [assumption].
12 (all x all y (mol(x,y) <-> in(x,y))) # label(non_clause). [assumption].
14 (all b (bond(b) -> (exists x exists y (atom(x) & atom(y) & mol(x,b) &
    ↪ mol(y,b))))) # label(non_clause) # label(goal). [goal].
16 -bond(x) | line(x). [clausify(11)].
17 bond(c3). [deny(14)].
22 -line(x) | point(f3(x)). [clausify(3)].
25 -line(x) | in(f3(x),x). [clausify(3)].
35 atom(x) | -point(x). [clausify(10)].
37 mol(x,y) | -in(x,y). [clausify(12)].
41 -atom(x) | -atom(y) | -mol(x,c3) | -mol(y,c3). [deny(14)].
42 line(c3). [resolve(17,a,16,a)].
43 -atom(x) | -mol(x,c3). [factor(41,a,b),merge(c)].
63 in(f3(c3),c3). [resolve(42,a,25,a)].
66 point(f3(c3)). [resolve(42,a,22,a)].
110 atom(f3(c3)). [resolve(66,a,35,b)].
137 -mol(f3(c3),c3). [ur(43,a,110,a)].
```

```
144 $F. [ur(37,a,137,a),unit_del(a,63)].

============================== end of proof ==========================
```

Proof D.3.8: $T_{nonisolated\_loopless} \cup \Pi_1 \models$ Axiom #8 in T_{most_graph}

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 17323 was started by cchui on MacBook-Pro.local,
Fri Oct 19 22:52:06 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.00 (+ 0.00) seconds.
% Length of proof is 12.
% Level of proof is 4.
% Maximum clause weight is 5.
% Given clauses 0.

6 (all p (point(p) -> -line(p))) # label(non_clause). [assumption].
9 (all p (point(p) -> (exists l (line(l) & in(p,l))))) # label(non_clause).
   ↪  [assumption].
10 (all x (atom(x) <-> point(x))) # label(non_clause). [assumption].
14 (all b (bond(b) -> (exists x exists y all z all u (atom(x) & atom(y) &
   ↪ atom(z) & atom(u) & mol(z,b) & mol(u,b) -> z = x | z = y | u = x | u
   ↪ = y)))) # label(non_clause) # label(goal). [goal].
16 -atom(x) | point(x). [clausify(10)].
17 atom(x). [deny(14)].
39 -point(x) | -line(x). [clausify(6)].
42 -point(x) | line(f4(x)). [clausify(9)].
51 point(x). [resolve(17,a,16,a)].
56 line(f4(x)). [back_unit_del(42),unit_del(a,51)].
58 -line(x). [back_unit_del(39),unit_del(a,51)].
59 $F. [resolve(58,a,56,a)].

============================== end of proof ==========================
```

Proof D.3.9: $T_{nonisolated\_loopless} \cup \Pi_1 \models$ Axiom #9 in T_{most_graph}

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 17329 was started by cchui on MacBook-Pro.local,
Fri Oct 19 22:52:19 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================
```

```
============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.10 (+ 0.00) seconds.
% Length of proof is 26.
% Level of proof is 5.
% Maximum clause weight is 13.
% Given clauses 118.

3 (all l (line(l) -> (exists x exists y (point(x) & point(y) & x != y & in(
   ↪ x,l) & in(y,l))))) # label(non_clause). [assumption].
10 (all x (atom(x) <-> point(x))) # label(non_clause). [assumption].
11 (all x (bond(x) <-> line(x))) # label(non_clause). [assumption].
12 (all x all y (mol(x,y) <-> in(x,y))) # label(non_clause). [assumption].
14 (all b (bond(b) -> (exists x exists y (atom(x) & atom(y) & mol(x,b) &
   ↪ mol(y,b) & y != x)))) # label(non_clause) # label(goal). [goal].
16 -bond(x) | line(x). [clausify(11)].
17 bond(c3). [deny(14)].
21 -line(x) | point(f2(x)). [clausify(3)].
22 -line(x) | point(f3(x)). [clausify(3)].
23 -line(x) | f3(x) != f2(x). [clausify(3)].
24 -line(x) | in(f2(x),x). [clausify(3)].
25 -line(x) | in(f3(x),x). [clausify(3)].
35 atom(x) | -point(x). [clausify(10)].
37 mol(x,y) | -in(x,y). [clausify(12)].
41 -atom(x) | -atom(y) | -mol(x,c3) | -mol(y,c3) | y = x. [deny(14)].
42 line(c3). [resolve(17,a,16,a)].
62 in(f3(c3),c3). [resolve(42,a,25,a)].
63 in(f2(c3),c3). [resolve(42,a,24,a)].
64 f3(c3) != f2(c3). [resolve(42,a,23,a)].
65 point(f3(c3)). [resolve(42,a,22,a)].
66 point(f2(c3)). [resolve(42,a,21,a)].
108 atom(f3(c3)). [resolve(65,a,35,b)].
113 atom(f2(c3)). [resolve(66,a,35,b)].
138 mol(f3(c3),c3). [resolve(62,a,37,b)].
141 mol(f2(c3),c3). [resolve(63,a,37,b)].
240 $F. [ur(41,a,113,a,b,108,a,d,138,a,e,64,a),unit_del(a,141)].

============================ end of proof ==========================
```

D.4 Proofs for T_{most_group} Verification

Let Δ_2 be the set of translation definitions:

$$(\forall x(point(x) \equiv atom(x))).$$
$$(\forall x(line(x) \equiv bond(x))).$$
$$(\forall x(plane(x) \equiv group(x))).$$
$$(\forall x \forall y\ (in(x,y) \equiv mol(x,y))).$$

$$T_{most\_group} \cup \Delta_2 \models T_{cycle\_path\_subgraph\_nonisolated}$$

The following proofs, as well as the input files, are available online in COLORE at `http://colore.oor.net/most/interprets`:

- $T_{most\_group} \cup \Delta_2 \models$ Axiom #1 in $T_{cycle\_path\_subgraph\_nonisolated}$
- $T_{most\_group} \cup \Delta_2 \models$ Axiom #2 in $T_{cycle\_path\_subgraph\_nonisolated}$
- $T_{most\_group} \cup \Delta_2 \models$ Axiom #3 in $T_{cycle\_path\_subgraph\_nonisolated}$
- $T_{most\_group} \cup \Delta_2 \models$ Axiom #4 in $T_{cycle\_path\_subgraph\_nonisolated}$
- $T_{most\_group} \cup \Delta_2 \models$ Axiom #5 in $T_{cycle\_path\_subgraph\_nonisolated}$
- $T_{most\_group} \cup \Delta_2 \models$ Axiom #6 in $T_{cycle\_path\_subgraph\_nonisolated}$
- $T_{most\_group} \cup \Delta_2 \models$ Axiom #7 in $T_{cycle\_path\_subgraph\_nonisolated}$
- $T_{most\_group} \cup \Delta_2 \models$ Axiom #8 in $T_{cycle\_path\_subgraph\_nonisolated}$
- $T_{most\_group} \cup \Delta_2 \models$ Axiom #9 in $T_{cycle\_path\_subgraph\_nonisolated}$
- $T_{most\_group} \cup \Delta_2 \models$ Axiom #10 in $T_{cycle\_path\_subgraph\_nonisolated}$
- $T_{most\_group} \cup \Delta_2 \models$ Axiom #11 in $T_{cycle\_path\_subgraph\_nonisolated}$
- $T_{most\_group} \cup \Delta_2 \models$ Axiom #12 in $T_{cycle\_path\_subgraph\_nonisolated}$
- $T_{most\_group} \cup \Delta_2 \models$ Axiom #13 in $T_{cycle\_path\_subgraph\_nonisolated}$
- $T_{most\_group} \cup \Delta_2 \models$ Axiom #14 in $T_{cycle\_path\_subgraph\_nonisolated}$
- $T_{most\_group} \cup \Delta_2 \models$ Axiom #15 in $T_{cycle\_path\_subgraph\_nonisolated}$
- $T_{most\_group} \cup \Delta_2 \models$ Axiom #16 in $T_{cycle\_path\_subgraph\_nonisolated}$
- $T_{most\_group} \cup \Delta_2 \models$ Axiom #17 in $T_{cycle\_path\_subgraph\_nonisolated}$
- $T_{most\_group} \cup \Delta_2 \models$ Axiom #18 in $T_{cycle\_path\_subgraph\_nonisolated}$

Proof D.4.1: $T_{most\_group} \cup \Delta_2 \models$ Axiom #1 in $T_{cycle\_path\_subgraph\_nonisolated}$

```
============================ prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3430 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:39:54 2018
```

```
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 2.56 (+ 0.01) seconds.
% Length of proof is 57.
% Level of proof is 3.
% Maximum clause weight is 64.
% Given clauses 64.

13 (all a1 all a2 all a3 all a4 all g all b1 all b2 all b3 (group(g) & atom
   ↪ (a1) & atom(a2) & atom(a3) & atom(a4) & mol(a1,g) & mol(a2,g) & mol(
   ↪ a3,g) & mol(a4,g) & bond(b1) & mol(b1,g) & mol(a1,b1) & mol(a2,b1) &
   ↪ bond(b2) & mol(b2,g) & mol(a1,b2) & mol(a3,b2) & bond(b3) & mol(b3,g)
   ↪  & mol(a1,b3) & mol(a4,b3) -> a4 = a3 | a2 = a3 | a4 = a2)) # label(
   ↪ non_clause). [assumption].
14 (all x (point(x) <-> atom(x))) # label(non_clause). [assumption].
15 (all x (line(x) <-> bond(x))) # label(non_clause). [assumption].
16 (all x (group(x) <-> plane(x))) # label(non_clause). [assumption].
18 (all x all y (in(x,y) <-> mol(x,y))) # label(non_clause). [assumption].
28 (all x all y all z all u all q all l1 all l2 all l3 (plane(q) & point(x)
   ↪  & point(y) & point(z) & point(u) & in(x,q) & in(y,q) & in(z,q) & in(
   ↪ u,q) & line(l1) & in(l1,q) & in(x,l1) & in(y,l1) & line(l2) & in(l2,q
   ↪ ) & in(x,l2) & in(z,l2) & line(l3) & in(l3,q) & in(x,l3) & in(u,l3)
   ↪ -> u = z | y = z | u = y)) # label(non_clause) # label(goal). [goal].
30 -point(x) | atom(x). [clausify(14)].
31 point(c1). [deny(28)].
32 point(c2). [deny(28)].
33 point(c3). [deny(28)].
34 point(c4). [deny(28)].
36 -line(x) | bond(x). [clausify(15)].
37 line(c6). [deny(28)].
38 line(c7). [deny(28)].
39 line(c8). [deny(28)].
40 group(x) | -plane(x). [clausify(16)].
42 plane(c5). [deny(28)].
46 -in(x,y) | mol(x,y). [clausify(18)].
47 in(c1,c5). [deny(28)].
48 in(c2,c5). [deny(28)].
49 in(c3,c5). [deny(28)].
50 in(c4,c5). [deny(28)].
51 in(c6,c5). [deny(28)].
52 in(c1,c6). [deny(28)].
53 in(c2,c6). [deny(28)].
54 in(c7,c5). [deny(28)].
55 in(c1,c7). [deny(28)].
56 in(c3,c7). [deny(28)].
57 in(c8,c5). [deny(28)].
58 in(c1,c8). [deny(28)].
```

```
59 in(c4,c8). [deny(28)].
84 -group(x) | -atom(y) | -atom(z) | -atom(u) | -atom(w) | -mol(y,x) | -mol
   ↪ (z,x) | -mol(u,x) | -mol(w,x) | -bond(v5) | -mol(v5,x) | -mol(y,v5) |
   ↪  -mol(z,v5) | -bond(v6) | -mol(v6,x) | -mol(y,v6) | -mol(u,v6) | -
   ↪ bond(v7) | -mol(v7,x) | -mol(y,v7) | -mol(w,v7) | w = u | u = z | w =
   ↪  z. [clausify(13)].
105 c4 != c3. [deny(28)].
106 c3 != c2. [deny(28)].
107 c4 != c2. [deny(28)].
108 atom(c1). [resolve(31,a,30,a)].
109 atom(c2). [resolve(32,a,30,a)].
110 atom(c3). [resolve(33,a,30,a)].
111 atom(c4). [resolve(34,a,30,a)].
112 bond(c6). [resolve(37,a,36,a)].
113 bond(c7). [resolve(38,a,36,a)].
114 bond(c8). [resolve(39,a,36,a)].
115 group(c5). [resolve(42,a,40,b)].
116 mol(c1,c5). [resolve(47,a,46,a)].
117 mol(c2,c5). [resolve(48,a,46,a)].
118 mol(c3,c5). [resolve(49,a,46,a)].
119 mol(c4,c5). [resolve(50,a,46,a)].
120 mol(c6,c5). [resolve(51,a,46,a)].
121 mol(c1,c6). [resolve(52,a,46,a)].
122 mol(c2,c6). [resolve(53,a,46,a)].
123 mol(c7,c5). [resolve(54,a,46,a)].
124 mol(c1,c7). [resolve(55,a,46,a)].
125 mol(c3,c7). [resolve(56,a,46,a)].
126 mol(c8,c5). [resolve(57,a,46,a)].
127 mol(c1,c8). [resolve(58,a,46,a)].
128 mol(c4,c8). [resolve(59,a,46,a)].
504 $F. [ur(84,a,115,a,b,108,a,c,110,a,d,109,a,e,111,a,f,116,a,g,118,a,h
   ↪ ,117,a,i,119,a,j,113,a,k,123,a,l,124,a,m,125,a,n,112,a,o,120,a,p,121,
   ↪ a,q,122,a,r,114,a,s,126,a,t,127,a,v,107,a,w,106,a(flip),x,105,a),
   ↪ unit_del(a,128)].

============================== end of proof ==========================
```

Proof D.4.2: $T_{most\_group} \cup \Delta_2 \models$ Axiom #2 in $T_{cycle\_path\_subgraph\_nonisolated}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3320 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:31:48 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 109.20 (+ 1.49) seconds.
% Length of proof is 20.
```

```
% Level of proof is 6.
% Maximum clause weight is 20.
% Given clauses 22687.

12 (all x all y all z all u (group(x) & end(y,x) & end(z,x) & end(u,x) -> y
   ↪  = z | y = u | z = u)) # label(non_clause). [assumption].
14 (all x all y (end(x,y) <-> atom(x) & group(y) & mol(x,y) & (all b1 all
   ↪ b2 all z all u (bond(b1) & bond(b2) & atom(z) & atom(u) & mol(x,b1) &
   ↪  mol(u,b1) & mol(x,b2) & mol(z,b2) & mol(u,y) & mol(z,y) -> z = u))))
   ↪  # label(non_clause). [assumption].
25 (all x all y (end(x,y) <-> planar_pendant(x,y))) # label(non_clause). [
   ↪ assumption].
36 (all q all x all y all z (planar_pendant(x,q) & planar_pendant(y,q) &
   ↪ planar_pendant(z,q) -> x = y | y = z | z = x)) # label(non_clause) #
   ↪ label(goal). [goal].
109 end(x,y) | -planar_pendant(x,y). [clausify(25)].
111 planar_pendant(c2,c1). [deny(36)].
112 planar_pendant(c3,c1). [deny(36)].
113 planar_pendant(c4,c1). [deny(36)].
139 -group(x) | -end(y,x) | -end(z,x) | -end(u,x) | z = y | u = y | u = z.
   ↪ [clausify(12)].
142 -end(x,y) | group(y). [clausify(14)].
177 c3 != c2. [deny(36)].
178 c4 != c3. [deny(36)].
179 c4 != c2. [deny(36)].
241 end(c2,c1). [resolve(111,a,109,b)].
242 end(c3,c1). [resolve(112,a,109,b)].
243 end(c4,c1). [resolve(113,a,109,b)].
477 group(c1). [resolve(241,a,142,a)].
479 -end(x,c1) | -end(y,c1) | y = x | c2 = x | c2 = y. [resolve(241,a,139,d
   ↪ ),unit_del(a,477)].
4109 -end(x,c1) | c4 = x | c2 = x. [resolve(479,a,243,a),flip(b),flip(c),
   ↪ unit_del(c,179)].
52345 $F. [resolve(4109,a,242,a),flip(b),unit_del(a,178),unit_del(b,177)].

============================== end of proof ==========================
```

Proof D.4.3: $T_{most\_group} \cup \Delta_2 \models$ Axiom #3 in $T_{cycle\_path\_subgraph\_nonisolated}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3337 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:34:21 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.94 (+ 0.02) seconds.
% Length of proof is 18.
```

```
% Level of proof is 5.
% Maximum clause weight is 6.
% Given clauses 406.

22 (all x (point(x) <-> atom(x))) # label(non_clause). [assumption].
23 (all x (line(x) <-> bond(x))) # label(non_clause). [assumption].
26 (all x all y (in(x,y) <-> mol(x,y))) # label(non_clause). [assumption].
32 (all x (atom(x) -> (exists b exists y (atom(y) & bond(b) & mol(x,b) &
   ↪ mol(y,b))))) # label(non_clause). [assumption].
36 (all p (point(p) -> (exists l (line(l) & in(p,l))))) # label(non_clause)
   ↪  # label(goal). [goal].
104 -point(x) | atom(x). [clausify(22)].
105 point(c1). [deny(36)].
106 line(x) | -bond(x). [clausify(23)].
108 -line(x) | -in(c1,x). [deny(36)].
155 in(x,y) | -mol(x,y). [clausify(26)].
162 -atom(x) | bond(f32(x)). [clausify(32)].
163 -atom(x) | mol(x,f32(x)). [clausify(32)].
237 atom(c1). [resolve(105,a,104,a)].
238 -in(c1,x) | -bond(x). [resolve(108,a,106,a)].
468 mol(c1,f32(c1)). [resolve(237,a,163,a)].
469 bond(f32(c1)). [resolve(237,a,162,a)].
809 in(c1,f32(c1)). [resolve(468,a,155,b)].
1585 $F. [resolve(809,a,238,a),unit_del(a,469)].

============================== end of proof ==========================
```

Proof D.4.4: $T_{most\_group} \cup \Delta_2 \models$ Axiom #4 in $T_{cycle\_path\_subgraph\_nonisolated}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3472 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:43:15 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.26 (+ 0.00) seconds.
% Length of proof is 30.
% Level of proof is 6.
% Maximum clause weight is 12.
% Given clauses 185.

8 (all b all g (bond(b) & group(g) & -mol(b,g) -> (exists a (atom(a) & mol(
   ↪ a,b) & -mol(a,g))))) # label(non_clause). [assumption].
9 (all x (point(x) <-> atom(x))) # label(non_clause). [assumption].
10 (all x (line(x) <-> bond(x))) # label(non_clause). [assumption].
11 (all x (group(x) <-> plane(x))) # label(non_clause). [assumption].
13 (all x all y (in(x,y) <-> mol(x,y))) # label(non_clause). [assumption].
```

```
23 (all l all q (line(l) & plane(q) & -in(l,q) -> (exists p (point(p) & in(
   ↪ p,l) & -in(p,q))))) # label(non_clause) # label(goal). [goal].
24 point(x) | -atom(x). [clausify(9)].
26 -point(x) | -in(x,c1) | in(x,c2). [deny(23)].
28 -line(x) | bond(x). [clausify(10)].
29 line(c1). [deny(23)].
30 group(x) | -plane(x). [clausify(11)].
32 plane(c2). [deny(23)].
46 -bond(x) | -group(y) | mol(x,y) | atom(f4(x,y)). [clausify(8)].
47 -bond(x) | -group(y) | mol(x,y) | mol(f4(x,y),x). [clausify(8)].
48 -bond(x) | -group(y) | mol(x,y) | -mol(f4(x,y),y). [clausify(8)].
49 -in(x,y) | mol(x,y). [clausify(13)].
50 in(x,y) | -mol(x,y). [clausify(13)].
71 -in(c1,c2). [deny(23)].
72 -in(x,c1) | in(x,c2) | -atom(x). [resolve(26,a,24,a)].
73 bond(c1). [resolve(29,a,28,a)].
74 group(c2). [resolve(32,a,30,b)].
77 -mol(c1,c2). [ur(50,a,71,a)].
89 -bond(x) | mol(x,c2) | mol(f4(x,c2),x). [resolve(74,a,47,b)].
90 -bond(x) | mol(x,c2) | atom(f4(x,c2)). [resolve(74,a,46,b)].
130 -mol(f4(c1,c2),c2). [ur(48,a,73,a,b,74,a,c,77,a)].
173 mol(f4(c1,c2),c1). [resolve(89,a,73,a),unit_del(a,77)].
217 atom(f4(c1,c2)). [resolve(90,a,73,a),unit_del(a,77)].
417 -in(f4(c1,c2),c2). [ur(49,b,130,a)].
430 -in(f4(c1,c2),c1). [ur(72,b,417,a,c,217,a)].
434 $F. [ur(50,a,430,a),unit_del(a,173)].

============================== end of proof ==========================
```

Proof D.4.5: $T_{most\_group} \cup \Delta_2 \models$ Axiom #5 in $T_{cycle\_path\_subgraph\_nonisolated}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3348 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:35:10 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 32.98 (+ 0.45) seconds.
% Length of proof is 36.
% Level of proof is 8.
% Maximum clause weight is 13.
% Given clauses 10161.

22 (all x (point(x) <-> atom(x))) # label(non_clause). [assumption].
23 (all x (line(x) <-> bond(x))) # label(non_clause). [assumption].
26 (all x all y (in(x,y) <-> mol(x,y))) # label(non_clause). [assumption].
33 (all b (bond(b) -> (exists x exists y (atom(x) & atom(y) & mol(x,b) &
```

```
   ↪ mol(y,b))))) # label(non_clause). [assumption].
35 (all b (bond(b) -> (exists x exists y (atom(x) & atom(y) & mol(x,b) &
   ↪ mol(y,b) & y != x)))) # label(non_clause). [assumption].
36 (all l (line(l) -> (exists x exists y (point(x) & point(y) & x != y & in
   ↪ (x,l) & in(y,l))))) # label(non_clause) # label(goal). [goal].
104 -line(x) | bond(x). [clausify(23)].
105 line(c1). [deny(36)].
152 point(x) | -atom(x). [clausify(22)].
154 in(x,y) | -mol(x,y). [clausify(26)].
164 -bond(x) | atom(f34(x)). [clausify(33)].
166 -bond(x) | mol(f34(x),x). [clausify(33)].
170 -bond(x) | atom(f38(x)). [clausify(35)].
171 -bond(x) | atom(f39(x)). [clausify(35)].
172 -bond(x) | mol(f38(x),x). [clausify(35)].
173 -bond(x) | mol(f39(x),x). [clausify(35)].
174 -bond(x) | f39(x) != f38(x). [clausify(35)].
175 -point(x) | -point(y) | y = x | -in(x,c1) | -in(y,c1). [deny(36)].
237 bond(c1). [resolve(105,a,104,a)].
466 f39(c1) != f38(c1). [resolve(237,a,174,a)].
467 mol(f39(c1),c1). [resolve(237,a,173,a)].
468 mol(f38(c1),c1). [resolve(237,a,172,a)].
469 atom(f39(c1)). [resolve(237,a,171,a)].
470 atom(f38(c1)). [resolve(237,a,170,a)].
472 mol(f34(c1),c1). [resolve(237,a,166,a)].
474 atom(f34(c1)). [resolve(237,a,164,a)].
481 point(f39(c1)). [resolve(469,a,152,b)].
495 point(f38(c1)). [resolve(470,a,152,b)].
523 point(f34(c1)). [resolve(474,a,152,b)].
816 in(f39(c1),c1). [resolve(467,a,154,b)].
1165 in(f38(c1),c1). [resolve(468,a,154,b)].
1825 in(f34(c1),c1). [resolve(472,a,154,b)].
2284 -point(x) | f39(c1) = x | -in(x,c1). [resolve(816,a,175,e),unit_del(b
   ↪ ,481)].
38869 f39(c1) = f34(c1). [resolve(2284,c,1825,a),unit_del(a,523)].
38871 f38(c1) = f34(c1). [resolve(2284,c,1165,a),rewrite([38869(5)]),flip(b
   ↪ ),unit_del(a,495)].
44890 $F. [back_rewrite(466),rewrite([38869(2),38871(4)]),xx(a)].

============================== end of proof ==========================
```

Proof D.4.6: $T_{most\_group} \cup \Delta_2 \models$ Axiom #6 in $T_{cycle\_path\_subgraph\_nonisolated}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3354 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:36:00 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================
```

```
% -------- Comments from original proof --------
% Proof 1 at 1.40 (+ 0.01) seconds.
% Length of proof is 11.
% Level of proof is 4.
% Maximum clause weight is 6.
% Given clauses 331.

26 (all x all y (in(x,y) <-> mol(x,y))) # label(non_clause). [assumption].
28 (all x all y (mol(x,y) -> mol(y,x))) # label(non_clause). [assumption].
36 (all x all y (in(x,y) -> in(y,x))) # label(non_clause) # label(goal). [
    ↪ goal].
152 -in(x,y) | mol(x,y). [clausify(26)].
153 in(x,y) | -mol(x,y). [clausify(26)].
155 -mol(x,y) | mol(y,x). [clausify(28)].
174 in(c1,c2). [deny(36)].
175 -in(c2,c1). [deny(36)].
453 mol(c1,c2). [resolve(174,a,152,a)].
759 mol(c2,c1). [resolve(453,a,155,a)].
1074 $F. [resolve(759,a,153,b),unit_del(a,175)].

============================== end of proof ==========================
```

Proof D.4.7: $T_{most\_group} \cup \Delta_2 \models$ Axiom #7 in $T_{cycle\_path\_subgraph\_nonisolated}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3360 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:36:16 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
    ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.42 (+ 0.01) seconds.
% Length of proof is 8.
% Level of proof is 3.
% Maximum clause weight is 6.
% Given clauses 41.

26 (all x all y (in(x,y) <-> mol(x,y))) # label(non_clause). [assumption].
27 (all x mol(x,x)) # label(non_clause). [assumption].
36 (all x (point(x) | line(x) | plane(x) -> in(x,x))) # label(non_clause) #
    ↪  label(goal). [goal].
156 in(x,y) | -mol(x,y). [clausify(26)].
157 mol(x,x). [clausify(27)].
177 -in(c1,c1). [deny(36)].
455 in(x,x). [resolve(157,a,156,b)].
456 $F. [resolve(455,a,177,a)].

============================== end of proof ==========================
```

Proof D.4.8: $T_{most\_group} \cup \Delta_2 \models$ Axiom #8 in $T_{cycle\_path\_subgraph\_nonisolated}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3366 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:36:31 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.45 (+ 0.01) seconds.
% Length of proof is 12.
% Level of proof is 3.
% Maximum clause weight is 4.
% Given clauses 117.

1 (all x (atom(x) -> -(bond(x) | group(x)))) # label(non_clause). [
   ↪ assumption].
22 (all x (point(x) <-> atom(x))) # label(non_clause). [assumption].
23 (all x (line(x) <-> bond(x))) # label(non_clause). [assumption].
36 (all p (point(p) -> -line(p))) # label(non_clause) # label(goal). [goal
   ↪ ].
104 -point(x) | atom(x). [clausify(22)].
105 point(c1). [deny(36)].
107 -line(x) | bond(x). [clausify(23)].
108 line(c1). [deny(36)].
115 -atom(x) | -bond(x). [clausify(1)].
237 atom(c1). [resolve(105,a,104,a)].
238 bond(c1). [resolve(108,a,107,a)].
488 $F. [resolve(238,a,115,b),unit_del(a,237)].

============================== end of proof ==========================
```

Proof D.4.9: $T_{most\_group} \cup \Delta_2 \models$ Axiom #9 in $T_{cycle\_path\_subgraph\_nonisolated}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3371 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:36:45 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.46 (+ 0.01) seconds.
% Length of proof is 12.
```

```
% Level of proof is 3.
% Maximum clause weight is 4.
% Given clauses 117.

1 (all x (atom(x) -> -(bond(x) | group(x)))) # label(non_clause). [
   ↪ assumption].
22 (all x (point(x) <-> atom(x))) # label(non_clause). [assumption].
24 (all x (group(x) <-> plane(x))) # label(non_clause). [assumption].
36 (all p (point(p) -> -plane(p))) # label(non_clause) # label(goal). [goal
   ↪ ].
104 -point(x) | atom(x). [clausify(22)].
105 point(c1). [deny(36)].
108 group(x) | -plane(x). [clausify(24)].
110 plane(c1). [deny(36)].
116 -atom(x) | -group(x). [clausify(1)].
237 atom(c1). [resolve(105,a,104,a)].
238 group(c1). [resolve(110,a,108,b)].
486 $F. [resolve(238,a,116,b),unit_del(a,237)].

============================== end of proof ==========================
```

Proof D.4.10: $T_{most\_group} \cup \Delta_2 \models$ Axiom #10 in $T_{cycle\_path\_subgraph\_nonisolated}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3377 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:37:02 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.44 (+ 0.01) seconds.
% Length of proof is 12.
% Level of proof is 3.
% Maximum clause weight is 4.
% Given clauses 117.

2 (all x (bond(x) -> -group(x))) # label(non_clause). [assumption].
23 (all x (line(x) <-> bond(x))) # label(non_clause). [assumption].
24 (all x (group(x) <-> plane(x))) # label(non_clause). [assumption].
36 (all p (plane(p) -> -line(p))) # label(non_clause) # label(goal). [goal
   ↪ ].
106 -line(x) | bond(x). [clausify(23)].
107 line(c1). [deny(36)].
108 group(x) | -plane(x). [clausify(24)].
110 plane(c1). [deny(36)].
117 -bond(x) | -group(x). [clausify(2)].
237 bond(c1). [resolve(107,a,106,a)].
238 group(c1). [resolve(110,a,108,b)].
```

```
482 $F. [resolve(238,a,117,b),unit_del(a,237)].

============================== end of proof ==========================
```

Proof D.4.11: $T_{most\_group} \cup \Delta_2 \models$ Axiom #11 in $T_{cycle\_path\_subgraph\_nonisolated}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3382 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:37:17 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.49 (+ 0.01) seconds.
% Length of proof is 15.
% Level of proof is 3.
% Maximum clause weight is 10.
% Given clauses 119.

22 (all x (point(x) <-> atom(x))) # label(non_clause). [assumption].
26 (all x all y (in(x,y) <-> mol(x,y))) # label(non_clause). [assumption].
29 (all x all y (mol(x,y) & atom(x) & atom(y) -> x = y)) # label(non_clause
   ↪ ). [assumption].
36 (all x all y (in(x,y) & point(x) & point(y) -> x = y)) # label(
   ↪ non_clause) # label(goal). [goal].
104 -point(x) | atom(x). [clausify(22)].
105 point(c1). [deny(36)].
106 point(c2). [deny(36)].
114 -in(x,y) | mol(x,y). [clausify(26)].
115 in(c1,c2). [deny(36)].
159 -mol(x,y) | -atom(x) | -atom(y) | y = x. [clausify(29)].
177 c2 != c1. [deny(36)].
239 atom(c1). [resolve(105,a,104,a)].
240 atom(c2). [resolve(106,a,104,a)].
241 mol(c1,c2). [resolve(115,a,114,a)].
612 $F. [resolve(241,a,159,a),unit_del(a,239),unit_del(b,240),unit_del(c
   ↪ ,177)].

============================== end of proof ==========================
```

Proof D.4.12: $T_{most\_group} \cup \Delta_2 \models$ Axiom #12 in $T_{cycle\_path\_subgraph\_nonisolated}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3388 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:37:34 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
```

```
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.46 (+ 0.01) seconds.
% Length of proof is 15.
% Level of proof is 3.
% Maximum clause weight is 10.
% Given clauses 119.

23 (all x (line(x) <-> bond(x))) # label(non_clause). [assumption].
26 (all x all y (in(x,y) <-> mol(x,y))) # label(non_clause). [assumption].
30 (all x all y (mol(x,y) & bond(x) & bond(y) -> x = y)) # label(non_clause
   ↪ ). [assumption].
36 (all x all y (in(x,y) & line(x) & line(y) -> x = y)) # label(non_clause)
   ↪  # label(goal). [goal].
106 -line(x) | bond(x). [clausify(23)].
107 line(c1). [deny(36)].
108 line(c2). [deny(36)].
114 -in(x,y) | mol(x,y). [clausify(26)].
115 in(c1,c2). [deny(36)].
160 -mol(x,y) | -bond(x) | -bond(y) | y = x. [clausify(30)].
177 c2 != c1. [deny(36)].
239 bond(c1). [resolve(107,a,106,a)].
240 bond(c2). [resolve(108,a,106,a)].
241 mol(c1,c2). [resolve(115,a,114,a)].
501 $F. [resolve(241,a,160,a),unit_del(a,239),unit_del(b,240),unit_del(c
   ↪ ,177)].

============================== end of proof ==========================
```

Proof D.4.13: $T_{most\_group} \cup \Delta_2 \models$ Axiom #13 in $T_{cycle\_path\_subgraph\_nonisolated}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3395 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:37:48 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.46 (+ 0.01) seconds.
% Length of proof is 15.
% Level of proof is 3.
% Maximum clause weight is 10.
% Given clauses 119.
```

```
3 (all x all y (mol(x,y) & group(x) & group(y) -> x = y)) # label(
   ↪ non_clause). [assumption].
24 (all x (group(x) <-> plane(x))) # label(non_clause). [assumption].
26 (all x all y (in(x,y) <-> mol(x,y))) # label(non_clause). [assumption].
36 (all x all y (in(x,y) & plane(x) & plane(y) -> x = y)) # label(
   ↪ non_clause) # label(goal). [goal].
107 group(x) | -plane(x). [clausify(24)].
109 plane(c1). [deny(36)].
110 plane(c2). [deny(36)].
114 -in(x,y) | mol(x,y). [clausify(26)].
115 in(c1,c2). [deny(36)].
119 -mol(x,y) | -group(x) | -group(y) | y = x. [clausify(3)].
177 c2 != c1. [deny(36)].
239 group(c1). [resolve(109,a,107,b)].
240 group(c2). [resolve(110,a,107,b)].
241 mol(c1,c2). [resolve(115,a,114,a)].
492 $F. [resolve(241,a,119,a),unit_del(a,239),unit_del(b,240),unit_del(c
   ↪ ,177)].

============================== end of proof ==========================
```

Proof D.4.14: $T_{most\_group} \cup \Delta_2 \models$ Axiom #14 in $T_{cycle\_path\_subgraph\_nonisolated}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3401 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:38:04 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 1.44 (+ 0.02) seconds.
% Length of proof is 18.
% Level of proof is 5.
% Maximum clause weight is 6.
% Given clauses 415.

5 (all x (atom(x) -> (exists y (group(y) & mol(x,y))))) # label(non_clause)
   ↪ . [assumption].
22 (all x (point(x) <-> atom(x))) # label(non_clause). [assumption].
24 (all x (group(x) <-> plane(x))) # label(non_clause). [assumption].
26 (all x all y (in(x,y) <-> mol(x,y))) # label(non_clause). [assumption].
36 (all p (point(p) -> (exists q (plane(q) & in(p,q))))) # label(non_clause
   ↪ ) # label(goal). [goal].
104 -point(x) | atom(x). [clausify(22)].
105 point(c1). [deny(36)].
109 -group(x) | plane(x). [clausify(24)].
110 -plane(x) | -in(c1,x). [deny(36)].
```

```
119 -atom(x) | group(f2(x)). [clausify(5)].
120 -atom(x) | mol(x,f2(x)). [clausify(5)].
155 in(x,y) | -mol(x,y). [clausify(26)].
237 atom(c1). [resolve(105,a,104,a)].
238 -in(c1,x) | -group(x). [resolve(110,a,109,b)].
478 mol(c1,f2(c1)). [resolve(237,a,120,a)].
479 group(f2(c1)). [resolve(237,a,119,a)].
1315 in(c1,f2(c1)). [resolve(478,a,155,b)].
1746 $F. [resolve(1315,a,238,a),unit_del(a,479)].

============================== end of proof ==========================
```

Proof D.4.15: $T_{most\_group} \cup \Delta_2 \models$ Axiom #15 in $T_{cycle\_path\_subgraph\_nonisolated}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3406 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:38:19 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 1.52 (+ 0.03) seconds.
% Length of proof is 18.
% Level of proof is 5.
% Maximum clause weight is 6.
% Given clauses 401.

22 (all x (point(x) <-> atom(x))) # label(non_clause). [assumption].
23 (all x (line(x) <-> bond(x))) # label(non_clause). [assumption].
26 (all x all y (in(x,y) <-> mol(x,y))) # label(non_clause). [assumption].
35 (all b (bond(b) -> (exists x exists y (atom(x) & atom(y) & mol(x,b) &
   ↪ mol(y,b) & y != x)))) # label(non_clause). [assumption].
36 (all l (line(l) -> (exists x (point(x) & in(x,l))))) # label(non_clause)
   ↪  # label(goal). [goal].
103 point(x) | -atom(x). [clausify(22)].
105 -point(x) | -in(x,c1). [deny(36)].
107 -line(x) | bond(x). [clausify(23)].
108 line(c1). [deny(36)].
155 in(x,y) | -mol(x,y). [clausify(26)].
172 -bond(x) | atom(f39(x)). [clausify(35)].
174 -bond(x) | mol(f39(x),x). [clausify(35)].
237 -in(x,c1) | -atom(x). [resolve(105,a,103,a)].
238 bond(c1). [resolve(108,a,107,a)].
468 mol(f39(c1),c1). [resolve(238,a,174,a)].
470 atom(f39(c1)). [resolve(238,a,172,a)].
809 in(f39(c1),c1). [resolve(468,a,155,b)].
2259 $F. [resolve(809,a,237,a),unit_del(a,470)].
```

```
============================== end of proof ==========================
```

Proof D.4.16: $T_{most\_group} \cup \Delta_2 \models$ Axiom #16 in $T_{cycle\_path\_subgraph\_nonisolated}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3479 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:43:57 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
    ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.01 (+ 0.00) seconds.
% Length of proof is 27.
% Level of proof is 3.
% Maximum clause weight is 26.
% Given clauses 42.

10 (all x (point(x) <-> atom(x))) # label(non_clause). [assumption].
11 (all x (line(x) <-> bond(x))) # label(non_clause). [assumption].
14 (all x all y (in(x,y) <-> mol(x,y))) # label(non_clause). [assumption].
19 (all x all y all z all b (atom(x) & atom(y) & atom(z) & bond(b) & mol(x,
    ↪ b) & mol(y,b) & mol(z,b) -> x = y | x = z | y = z)) # label(
    ↪ non_clause). [assumption].
24 (all x all y all z all l (point(x) & point(y) & point(z) & line(l) & in(
    ↪ x,l) & in(y,l) & in(z,l) -> z = x | z = y | x = y)) # label(
    ↪ non_clause) # label(goal). [goal].
26 -point(x) | atom(x). [clausify(10)].
27 point(c1). [deny(24)].
28 point(c2). [deny(24)].
29 point(c3). [deny(24)].
31 -line(x) | bond(x). [clausify(11)].
32 line(c4). [deny(24)].
38 -in(x,y) | mol(x,y). [clausify(14)].
39 in(c1,c4). [deny(24)].
40 in(c2,c4). [deny(24)].
41 in(c3,c4). [deny(24)].
61 -atom(x) | -atom(y) | -atom(z) | -bond(u) | -mol(x,u) | -mol(y,u) | -mol
    ↪ (z,u) | y = x | z = x | z = y. [clausify(19)].
77 c3 != c1. [deny(24)].
78 c3 != c2. [deny(24)].
79 c2 != c1. [deny(24)].
80 atom(c1). [resolve(27,a,26,a)].
81 atom(c2). [resolve(28,a,26,a)].
82 atom(c3). [resolve(29,a,26,a)].
83 bond(c4). [resolve(32,a,31,a)].
84 mol(c1,c4). [resolve(39,a,38,a)].
85 mol(c2,c4). [resolve(40,a,38,a)].
86 mol(c3,c4). [resolve(41,a,38,a)].
```

```
135 $F. [ur(61,a,82,a,b,80,a,c,81,a,d,83,a,f,84,a,g,85,a,h,77,a(flip),i,78,
    ↪ a(flip),j,79,a),unit_del(a,86)].

============================== end of proof ==========================
```

Proof D.4.17: $T_{most\_group} \cup \Delta_2 \models$ Axiom #17 in $T_{cycle\_path\_subgraph\_nonisolated}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3412 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:38:43 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
    ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 10.02 (+ 0.16) seconds.
% Length of proof is 26.
% Level of proof is 5.
% Maximum clause weight is 15.
% Given clauses 2552.

8 (all x all y all z (atom(x) & bond(y) & group(z) & mol(x,y) & mol(y,z) ->
    ↪  mol(x,z))) # label(non_clause). [assumption].
22 (all x (point(x) <-> atom(x))) # label(non_clause). [assumption].
23 (all x (line(x) <-> bond(x))) # label(non_clause). [assumption].
24 (all x (group(x) <-> plane(x))) # label(non_clause). [assumption].
26 (all x all y (in(x,y) <-> mol(x,y))) # label(non_clause). [assumption].
36 (all x all y all z (plane(x) & line(y) & point(z) & in(z,y) & in(y,x) ->
    ↪  in(z,x))) # label(non_clause) # label(goal). [goal].
104 -point(x) | atom(x). [clausify(22)].
105 point(c3). [deny(36)].
107 -line(x) | bond(x). [clausify(23)].
108 line(c2). [deny(36)].
109 group(x) | -plane(x). [clausify(24)].
111 plane(c1). [deny(36)].
130 -atom(x) | -bond(y) | -group(z) | -mol(x,y) | -mol(y,z) | mol(x,z). [
    ↪ clausify(8)].
155 -in(x,y) | mol(x,y). [clausify(26)].
156 in(x,y) | -mol(x,y). [clausify(26)].
177 in(c3,c2). [deny(36)].
178 in(c2,c1). [deny(36)].
179 -in(c3,c1). [deny(36)].
241 atom(c3). [resolve(105,a,104,a)].
242 bond(c2). [resolve(108,a,107,a)].
243 group(c1). [resolve(111,a,109,b)].
460 mol(c3,c2). [resolve(177,a,155,a)].
461 mol(c2,c1). [resolve(178,a,155,a)].
802 -group(x) | -mol(c2,x) | mol(c3,x). [resolve(460,a,130,d),unit_del(a
    ↪ ,241),unit_del(b,242)].
```

```
27096 mol(c3,c1). [resolve(802,b,461,a),unit_del(a,243)].
27149 $F. [resolve(27096,a,156,b),unit_del(a,179)].

============================== end of proof ==========================
```

Proof D.4.18: $T_{most\_group} \cup \Delta_2 \models$ Axiom #18 in $T_{cycle\_path\_subgraph\_nonisolated}$

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3421 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:39:12 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.82 (+ 0.01) seconds.
% Length of proof is 18.
% Level of proof is 5.
% Maximum clause weight is 6.
% Given clauses 367.

9 (all x (group(x) -> (exists y (atom(y) & mol(y,x))))) # label(non_clause)
   ↪ . [assumption].
22 (all x (point(x) <-> atom(x))) # label(non_clause). [assumption].
24 (all x (group(x) <-> plane(x))) # label(non_clause). [assumption].
26 (all x all y (in(x,y) <-> mol(x,y))) # label(non_clause). [assumption].
36 (all q (plane(q) -> (exists p (point(p) & in(p,q))))) # label(non_clause
   ↪ ) # label(goal). [goal].
103 point(x) | -atom(x). [clausify(22)].
105 -point(x) | -in(x,c1). [deny(36)].
108 group(x) | -plane(x). [clausify(24)].
110 plane(c1). [deny(36)].
130 -group(x) | atom(f6(x)). [clausify(9)].
131 -group(x) | mol(f6(x),x). [clausify(9)].
155 in(x,y) | -mol(x,y). [clausify(26)].
237 -in(x,c1) | -atom(x). [resolve(105,a,103,a)].
238 group(c1). [resolve(110,a,108,b)].
469 mol(f6(c1),c1). [resolve(238,a,131,a)].
470 atom(f6(c1)). [resolve(238,a,130,a)].
546 in(f6(c1),c1). [resolve(469,a,155,b)].
806 $F. [resolve(546,a,237,a),unit_del(a,470)].

============================== end of proof ==========================
```

D.5 Proofs for T_{most_group} Reduction

Let Π_2 be the set of translation definitions:

$$(\forall x(atom(x) \equiv\ point(x))).$$
$$(\forall x(bond(x) \equiv\ line(x))).$$
$$(\forall x(group(x) \equiv\ plane(x))).$$
$$\begin{aligned}(\forall x \forall y(end(x,y) \equiv\ & point(x) \wedge plane(y) \wedge in(x,y) \wedge (\forall l_1 \forall l_2 \forall w \forall z(line(l_1) \wedge \\ & line(l_2) \wedge point(w) \wedge point(z) \wedge in(x,l_1) \wedge in(z,l_1) \wedge in(x,l_2) \wedge \\ & in(w,l_2) \wedge in(z,y) \wedge in(w,y) \supset\ w = z)))).\end{aligned}$$
$$\begin{aligned}(\forall x \forall y(fork(x) \equiv\ & point(x) \wedge (\exists l_1 \exists l_2 \exists l_3 \exists p_1 \exists p_2 \exists p_3(point(p_1) \wedge point(p_2) \wedge \\ & point(p_3) \wedge line(l_1) \wedge line(l_2) \wedge line(l_3) \wedge p_1 \neq p_2 \wedge p_2 \neq p_3 \wedge \\ & p_1 \neq p_3 \wedge l_1 \neq l_2 \wedge l_2 \neq l_3 \wedge l_1 \neq l_3 \wedge in(x,l_1) \wedge in(p_1,l_1) \wedge \\ & in(x,l_2) \wedge in(p_2,l_2) \wedge in(x,l_3) \wedge in(p_3,l_3))))).\end{aligned}$$

$$(\forall x \forall y(mol(x,y) \equiv\ in(x,y))).$$

$$T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models T_{most\_group}$$

The following proofs, as well as the input files, are available online in COLORE at `http://colore.oor.net/most/interprets`:

- $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #1 in T_{most_group}
- $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #2 in T_{most_group}
- $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #3 in T_{most_group}
- $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #4 in T_{most_group}
- $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #5 in T_{most_group}
- $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #6 in T_{most_group}
- $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #7 in T_{most_group}
- $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #8 in T_{most_group}
- $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #9 in T_{most_group}
- $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #10 in T_{most_group}
- $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #11 in T_{most_group}
- $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #12 in T_{most_group}
- $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #13 in T_{most_group}

Proof D.5.1: $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #1 in T_{most_group}

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3521 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:49:02 2018
```

```
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
    ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.04 (+ 0.00) seconds.
% Length of proof is 19.
% Level of proof is 4.
% Maximum clause weight is 4.
% Given clauses 31.

8 (all p (point(p) -> -line(p))) # label(non_clause). [assumption].
9 (all p (point(p) -> -plane(p))) # label(non_clause). [assumption].
19 (all x (atom(x) <-> point(x))) # label(non_clause). [assumption].
20 (all x (bond(x) <-> line(x))) # label(non_clause). [assumption].
21 (all x (group(x) <-> plane(x))) # label(non_clause). [assumption].
24 (all x (atom(x) -> -(bond(x) | group(x)))) # label(non_clause) # label(
    ↪ goal). [goal].
26 -atom(x) | point(x). [clausify(19)].
27 atom(c1). [deny(24)].
29 -bond(x) | line(x). [clausify(20)].
30 bond(c1) | group(c1). [deny(24)].
32 -group(x) | plane(x). [clausify(21)].
33 group(c1) | line(c1). [resolve(30,a,29,a)].
54 -point(x) | -line(x). [clausify(8)].
55 -point(x) | -plane(x). [clausify(9)].
68 point(c1). [resolve(27,a,26,a)].
69 line(c1) | plane(c1). [resolve(33,a,32,a)].
84 -plane(c1). [ur(55,a,68,a)].
85 -line(c1). [ur(54,a,68,a)].
86 $F. [back_unit_del(69),unit_del(a,85),unit_del(b,84)].

============================== end of proof ==========================
```

Proof D.5.2: $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #2 in T_{most_group}

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3527 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:49:27 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
    ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.05 (+ 0.00) seconds.
```

```
% Length of proof is 12.
% Level of proof is 3.
% Maximum clause weight is 4.
% Given clauses 31.

10 (all p (plane(p) -> -line(p))) # label(non_clause). [assumption].
20 (all x (bond(x) <-> line(x))) # label(non_clause). [assumption].
21 (all x (group(x) <-> plane(x))) # label(non_clause). [assumption].
24 (all x (bond(x) -> -group(x))) # label(non_clause) # label(goal). [goal
   ↪ ].
28 -bond(x) | line(x). [clausify(20)].
29 bond(c1). [deny(24)].
31 -group(x) | plane(x). [clausify(21)].
32 group(c1). [deny(24)].
55 -plane(x) | -line(x). [clausify(10)].
67 line(c1). [resolve(29,a,28,a)].
68 plane(c1). [resolve(32,a,31,a)].
87 $F. [ur(55,b,67,a),unit_del(a,68)].

============================== end of proof ==========================
```

Proof D.5.3: $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #3 in T_{most_group}

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3536 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:50:29 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.05 (+ 0.00) seconds.
% Length of proof is 15.
% Level of proof is 3.
% Maximum clause weight is 10.
% Given clauses 33.

13 (all x all y (in(x,y) & plane(x) & plane(y) -> x = y)) # label(
   ↪ non_clause). [assumption].
21 (all x (group(x) <-> plane(x))) # label(non_clause). [assumption].
23 (all x all y (mol(x,y) <-> in(x,y))) # label(non_clause). [assumption].
24 (all x all y (mol(x,y) & group(x) & group(y) -> x = y)) # label(
   ↪ non_clause) # label(goal). [goal].
30 -group(x) | plane(x). [clausify(21)].
31 group(c1). [deny(24)].
32 group(c2). [deny(24)].
36 -mol(x,y) | in(x,y). [clausify(23)].
37 mol(c1,c2). [deny(24)].
59 -in(x,y) | -plane(x) | -plane(y) | y = x. [clausify(13)].
```

```
68 c2 != c1. [deny(24)].
69 plane(c1). [resolve(31,a,30,a)].
70 plane(c2). [resolve(32,a,30,a)].
71 in(c1,c2). [resolve(37,a,36,a)].
95 $F. [ur(59,b,69,a,c,70,a,d,68,a),unit_del(a,71)].

============================== end of proof ==========================
```

Proof D.5.4: $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #4 in T_{most_group}

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3543 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:50:47 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
    ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.20 (+ 0.00) seconds.
% Length of proof is 18.
% Level of proof is 5.
% Maximum clause weight is 6.
% Given clauses 44.

18 (all q (plane(q) -> (exists p (point(p) & in(p,q))))) # label(non_clause
    ↪ ). [assumption].
19 (all x (atom(x) <-> point(x))) # label(non_clause). [assumption].
21 (all x (group(x) <-> plane(x))) # label(non_clause). [assumption].
23 (all x all y (mol(x,y) <-> in(x,y))) # label(non_clause). [assumption].
24 (all x (group(x) -> (exists a (atom(a) & mol(a,x))))) # label(non_clause
    ↪ ) # label(goal). [goal].
25 atom(x) | -point(x). [clausify(19)].
27 -atom(x) | -mol(x,c1). [deny(24)].
31 -group(x) | plane(x). [clausify(21)].
32 group(c1). [deny(24)].
63 -plane(x) | point(f7(x)). [clausify(18)].
64 -plane(x) | in(f7(x),x). [clausify(18)].
66 mol(x,y) | -in(x,y). [clausify(23)].
67 -mol(x,c1) | -point(x). [resolve(27,a,25,a)].
68 plane(c1). [resolve(32,a,31,a)].
78 in(f7(c1),c1). [resolve(68,a,64,a)].
79 point(f7(c1)). [resolve(68,a,63,a)].
106 mol(f7(c1),c1). [resolve(78,a,66,b)].
125 $F. [ur(67,b,79,a),unit_del(a,106)].

============================== end of proof ==========================
```

Proof D.5.5: $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #5 in T_{most_group}

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3550 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:51:03 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.12 (+ 0.00) seconds.
% Length of proof is 18.
% Level of proof is 5.
% Maximum clause weight is 6.
% Given clauses 44.

14 (all p (point(p) -> (exists q (plane(q) & in(p,q))))) # label(non_clause
   ↪ ). [assumption].
19 (all x (atom(x) <-> point(x))) # label(non_clause). [assumption].
21 (all x (group(x) <-> plane(x))) # label(non_clause). [assumption].
23 (all x all y (mol(x,y) <-> in(x,y))) # label(non_clause). [assumption].
24 (all x (atom(x) -> (exists y (group(y) & mol(x,y))))) # label(non_clause
   ↪ ) # label(goal). [goal].
26 -atom(x) | point(x). [clausify(19)].
27 atom(c1). [deny(24)].
30 group(x) | -plane(x). [clausify(21)].
32 -group(x) | -mol(c1,x). [deny(24)].
57 -point(x) | plane(f5(x)). [clausify(14)].
58 -point(x) | in(x,f5(x)). [clausify(14)].
66 mol(x,y) | -in(x,y). [clausify(23)].
67 point(c1). [resolve(27,a,26,a)].
68 -mol(c1,x) | -plane(x). [resolve(32,a,30,a)].
78 in(c1,f5(c1)). [resolve(67,a,58,a)].
79 plane(f5(c1)). [resolve(67,a,57,a)].
107 mol(c1,f5(c1)). [resolve(78,a,66,b)].
129 $F. [ur(68,b,79,a),unit_del(a,107)].

============================== end of proof ==========================
```

Proof D.5.6: $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #6 in T_{most_group}

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3702 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:57:21 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================
```

```
============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 6.70 (+ 0.12) seconds.
% Length of proof is 54.
% Level of proof is 11.
% Maximum clause weight is 19.
% Given clauses 1950.

3 (all p (point(p) -> (exists l (line(l) & in(p,l))))) # label(non_clause).
    ↪  [assumption].
5 (all l (line(l) -> (exists x exists y (point(x) & point(y) & x != y & in(
    ↪ x,l) & in(y,l))))) # label(non_clause). [assumption].
14 (all p (point(p) -> (exists q (plane(q) & in(p,q))))) # label(non_clause
    ↪ ). [assumption].
27 (all x (atom(x) <-> point(x))) # label(non_clause). [assumption].
28 (all x (bond(x) <-> line(x))) # label(non_clause). [assumption].
29 (all x (group(x) <-> plane(x))) # label(non_clause). [assumption].
31 (all x all y (mol(x,y) <-> in(x,y))) # label(non_clause). [assumption].
32 (all x (atom(x) -> (exists b exists y exists z (bond(b) & group(y) &
    ↪ atom(z) & x != z & mol(x,b) & mol(z,b) & mol(z,y))))) # label(
    ↪ non_clause) # label(goal). [goal].
95 atom(x) | -point(x). [clausify(27)].
96 -atom(x) | point(x). [clausify(27)].
97 atom(c1). [deny(32)].
98 -bond(x) | -group(y) | -atom(z) | z = c1 | -mol(c1,x) | -mol(z,x) | -mol
    ↪ (z,y). [deny(32)].
99 bond(x) | -line(x). [clausify(28)].
101 -bond(x) | -group(y) | z = c1 | -mol(c1,x) | -mol(z,x) | -mol(z,y) | -
    ↪ point(z). [resolve(98,c,95,a)].
103 group(x) | -plane(x). [clausify(29)].
105 -group(x) | y = c1 | -mol(c1,z) | -mol(y,z) | -mol(y,x) | -point(y) | -
    ↪ line(z). [resolve(101,a,99,a)].
111 -point(x) | line(f1(x)). [clausify(3)].
112 -point(x) | in(x,f1(x)). [clausify(3)].
116 -line(x) | point(f3(x)). [clausify(5)].
117 -line(x) | point(f4(x)). [clausify(5)].
118 -line(x) | f4(x) != f3(x). [clausify(5)].
119 -line(x) | in(f3(x),x). [clausify(5)].
120 -line(x) | in(f4(x),x). [clausify(5)].
131 -point(x) | plane(f5(x)). [clausify(14)].
132 -point(x) | in(x,f5(x)). [clausify(14)].
158 mol(x,y) | -in(x,y). [clausify(31)].
224 point(c1). [resolve(97,a,96,a)].
225 x = c1 | -mol(c1,y) | -mol(x,y) | -mol(x,z) | -point(x) | -line(y) | -
    ↪ plane(z). [resolve(105,a,103,a)].
226 c1 = x | -mol(c1,y) | -mol(x,y) | -mol(x,z) | -point(x) | -line(y) | -
    ↪ plane(z). [copy(225),flip(a)].
266 in(c1,f1(c1)). [resolve(224,a,112,a)].
267 line(f1(c1)). [resolve(224,a,111,a)].
447 in(f4(f1(c1)),f1(c1)). [resolve(267,a,120,a)].
448 in(f3(f1(c1)),f1(c1)). [resolve(267,a,119,a)].
450 point(f4(f1(c1))). [resolve(267,a,117,a)].
451 point(f3(f1(c1))). [resolve(267,a,116,a)].
```

```
588 mol(c1,f1(c1)). [resolve(266,a,158,b)].
646 in(f4(f1(c1)),f5(f4(f1(c1)))). [resolve(450,a,132,a)].
647 plane(f5(f4(f1(c1)))). [resolve(450,a,131,a)].
653 in(f3(f1(c1)),f5(f3(f1(c1)))). [resolve(451,a,132,a)].
654 plane(f5(f3(f1(c1)))). [resolve(451,a,131,a)].
659 line(f1(f3(f1(c1)))). [resolve(451,a,111,a)].
660 c1 = x | -mol(x,f1(c1)) | -mol(x,y) | -point(x) | -plane(y). [resolve
   ↪ (588,a,226,b),unit_del(e,267)].
956 point(f3(f1(f3(f1(c1))))). [resolve(659,a,116,a)].
1338 mol(f4(f1(c1)),f1(c1)). [resolve(447,a,158,b)].
1519 mol(f3(f1(c1)),f1(c1)). [resolve(448,a,158,b)].
1689 line(f1(f3(f1(f3(f1(c1)))))). [resolve(956,a,111,a)].
2012 point(f3(f1(f3(f1(f3(f1(c1))))))). [resolve(1689,a,116,a)].
2566 mol(f4(f1(c1)),f5(f4(f1(c1)))). [resolve(646,a,158,b)].
2819 mol(f3(f1(c1)),f5(f3(f1(c1)))). [resolve(653,a,158,b)].
3593 line(f1(f3(f1(f3(f1(f3(f1(c1)))))))). [resolve(2012,a,111,a)].
6116 f4(f1(f3(f1(f3(f1(f3(f1(c1))))))) != f3(f1(f3(f1(f3(f1(f3(f1(c1))))))
   ↪ )). [resolve(3593,a,118,a)].
14105 f3(f1(c1)) = c1. [resolve(660,c,2819,a),flip(a),unit_del(b,1519),
   ↪ unit_del(c,451),unit_del(d,654)].
14107 f4(f1(c1)) = c1. [resolve(660,c,2566,a),flip(a),unit_del(b,1338),
   ↪ unit_del(c,450),unit_del(d,647)].
14112 $F. [back_rewrite(6116),rewrite([14105(3),14105(3),14105(3),14107(3)
   ↪ ,14105(4),14105(4),14105(4),14105(4)]),xx(a)].

============================== end of proof ==========================
```

Proof D.5.7: $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #7 in T_{most_group}

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3640 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:53:53 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.26 (+ 0.00) seconds.
% Length of proof is 25.
% Level of proof is 3.
% Maximum clause weight is 15.
% Given clauses 57.

17 (all x all y all z (plane(x) & line(y) & point(z) & in(z,y) & in(y,x) ->
   ↪  in(z,x))) # label(non_clause). [assumption].
19 (all x (atom(x) <-> point(x))) # label(non_clause). [assumption].
20 (all x (bond(x) <-> line(x))) # label(non_clause). [assumption].
21 (all x (group(x) <-> plane(x))) # label(non_clause). [assumption].
23 (all x all y (mol(x,y) <-> in(x,y))) # label(non_clause). [assumption].
```

```
24 (all b all g all x all y (atom(x) & atom(y) & x != y & bond(b) & group(g
   ↪ ) & mol(x,b) & mol(y,b) & mol(b,g) -> mol(x,g))) # label(non_clause)
   ↪ # label(goal). [goal].
26 -atom(x) | point(x). [clausify(19)].
27 atom(c3). [deny(24)].
30 -bond(x) | line(x). [clausify(20)].
31 bond(c1). [deny(24)].
33 -group(x) | plane(x). [clausify(21)].
34 group(c2). [deny(24)].
64 -plane(x) | -line(y) | -point(z) | -in(z,y) | -in(y,x) | in(z,x). [
   ↪ clausify(17)].
67 -mol(x,y) | in(x,y). [clausify(23)].
68 mol(x,y) | -in(x,y). [clausify(23)].
70 mol(c3,c1). [deny(24)].
72 mol(c1,c2). [deny(24)].
73 -mol(c3,c2). [deny(24)].
74 point(c3). [resolve(27,a,26,a)].
76 line(c1). [resolve(31,a,30,a)].
77 plane(c2). [resolve(34,a,33,a)].
87 in(c3,c1). [resolve(70,a,67,a)].
89 in(c1,c2). [resolve(72,a,67,a)].
90 -in(c3,c2). [ur(68,a,73,a)].
201 $F. [ur(64,b,76,a,c,74,a,d,87,a,e,89,a,f,90,a),unit_del(a,77)].

============================== end of proof ==========================
```

Proof D.5.8: $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #8 in T_{most_group}

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3647 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:54:10 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.18 (+ 0.00) seconds.
% Length of proof is 25.
% Level of proof is 3.
% Maximum clause weight is 15.
% Given clauses 54.

17 (all x all y all z (plane(x) & line(y) & point(z) & in(z,y) & in(y,x) ->
   ↪  in(z,x))) # label(non_clause). [assumption].
19 (all x (atom(x) <-> point(x))) # label(non_clause). [assumption].
20 (all x (bond(x) <-> line(x))) # label(non_clause). [assumption].
21 (all x (group(x) <-> plane(x))) # label(non_clause). [assumption].
23 (all x all y (mol(x,y) <-> in(x,y))) # label(non_clause). [assumption].
24 (all x all y all z (atom(x) & bond(y) & group(z) & mol(x,y) & mol(y,z)
```

```
   ↪ -> mol(x,z))) # label(non_clause) # label(goal). [goal].
26 -atom(x) | point(x). [clausify(19)].
27 atom(c1). [deny(24)].
29 -bond(x) | line(x). [clausify(20)].
30 bond(c2). [deny(24)].
32 -group(x) | plane(x). [clausify(21)].
33 group(c3). [deny(24)].
63 -plane(x) | -line(y) | -point(z) | -in(z,y) | -in(y,x) | in(z,x). [
   ↪ clausify(17)].
66 -mol(x,y) | in(x,y). [clausify(23)].
67 mol(x,y) | -in(x,y). [clausify(23)].
68 mol(c1,c2). [deny(24)].
69 mol(c2,c3). [deny(24)].
70 -mol(c1,c3). [deny(24)].
71 point(c1). [resolve(27,a,26,a)].
72 line(c2). [resolve(30,a,29,a)].
73 plane(c3). [resolve(33,a,32,a)].
83 in(c1,c2). [resolve(68,a,66,a)].
84 in(c2,c3). [resolve(69,a,66,a)].
85 -in(c1,c3). [ur(67,a,70,a)].
174 $F. [ur(63,b,72,a,c,71,a,d,83,a,e,84,a,f,85,a),unit_del(a,73)].

============================== end of proof ==========================
```

Proof D.5.9: $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #9 in T_{most_group}

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3653 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:54:25 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.20 (+ 0.00) seconds.
% Length of proof is 18.
% Level of proof is 5.
% Maximum clause weight is 6.
% Given clauses 44.

18 (all q (plane(q) -> (exists p (point(p) & in(p,q))))) # label(non_clause
   ↪ ). [assumption].
19 (all x (atom(x) <-> point(x))) # label(non_clause). [assumption].
21 (all x (group(x) <-> plane(x))) # label(non_clause). [assumption].
23 (all x all y (mol(x,y) <-> in(x,y))) # label(non_clause). [assumption].
24 (all x (group(x) -> (exists y (atom(y) & mol(y,x))))) # label(non_clause
   ↪ ) # label(goal). [goal].
25 atom(x) | -point(x). [clausify(19)].
27 -atom(x) | -mol(x,c1). [deny(24)].
```

```
31 -group(x) | plane(x). [clausify(21)].
32 group(c1). [deny(24)].
63 -plane(x) | point(f7(x)). [clausify(18)].
64 -plane(x) | in(f7(x),x). [clausify(18)].
66 mol(x,y) | -in(x,y). [clausify(23)].
67 -mol(x,c1) | -point(x). [resolve(27,a,25,a)].
68 plane(c1). [resolve(32,a,31,a)].
78 in(f7(c1),c1). [resolve(68,a,64,a)].
79 point(f7(c1)). [resolve(68,a,63,a)].
106 mol(f7(c1),c1). [resolve(78,a,66,b)].
125 $F. [ur(67,b,79,a),unit_del(a,106)].

============================== end of proof ==========================
```

Proof D.5.10: $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #10 in T_{most_group}

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3662 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:54:42 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.05 (+ 0.00) seconds.
% Length of proof is 15.
% Level of proof is 3.
% Maximum clause weight is 10.
% Given clauses 40.

11 (all x all y (in(x,y) & point(x) & point(y) -> x = y)) # label(
   ↪ non_clause). [assumption].
19 (all x (atom(x) <-> point(x))) # label(non_clause). [assumption].
23 (all x all y (mol(x,y) <-> in(x,y))) # label(non_clause). [assumption].
24 (all x all y all b all g (atom(x) & atom(y) & bond(b) & group(g) & x !=
   ↪ y & mol(x,y) & mol(y,g) & mol(x,b) & mol(y,b) -> mol(b,g))) # label(
   ↪ non_clause) # label(goal). [goal].
26 -atom(x) | point(x). [clausify(19)].
27 atom(c1). [deny(24)].
28 atom(c2). [deny(24)].
56 -in(x,y) | -point(x) | -point(y) | y = x. [clausify(11)].
67 -mol(x,y) | in(x,y). [clausify(23)].
69 c2 != c1. [deny(24)].
70 mol(c1,c2). [deny(24)].
75 point(c1). [resolve(27,a,26,a)].
76 point(c2). [resolve(28,a,26,a)].
88 in(c1,c2). [resolve(70,a,67,a)].
105 $F. [ur(56,b,75,a,c,76,a,d,69,a),unit_del(a,88)].
```

```
============================== end of proof ==========================
```

Proof D.5.11: $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #11 in T_{most_group}

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3667 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:54:58 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
    ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 9.37 (+ 0.02) seconds.
% Length of proof is 30.
% Level of proof is 6.
% Maximum clause weight is 12.
% Given clauses 127.

4 (all l all q (line(l) & plane(q) & -in(l,q) -> (exists p (point(p) & in(p
    ↪ ,l) & -in(p,q))))) # label(non_clause). [assumption].
19 (all x (atom(x) <-> point(x))) # label(non_clause). [assumption].
20 (all x (bond(x) <-> line(x))) # label(non_clause). [assumption].
21 (all x (group(x) <-> plane(x))) # label(non_clause). [assumption].
23 (all x all y (mol(x,y) <-> in(x,y))) # label(non_clause). [assumption].
24 (all b all g (bond(b) & group(g) & -mol(b,g) -> (exists a (atom(a) & mol
    ↪ (a,b) & -mol(a,g))))) # label(non_clause) # label(goal). [goal].
25 atom(x) | -point(x). [clausify(19)].
27 -atom(x) | -mol(x,c1) | mol(x,c2). [deny(24)].
29 -bond(x) | line(x). [clausify(20)].
30 bond(c1). [deny(24)].
32 -group(x) | plane(x). [clausify(21)].
33 group(c2). [deny(24)].
40 -line(x) | -plane(y) | in(x,y) | point(f2(x,y)). [clausify(4)].
41 -line(x) | -plane(y) | in(x,y) | in(f2(x,y),x). [clausify(4)].
42 -line(x) | -plane(y) | in(x,y) | -in(f2(x,y),y). [clausify(4)].
66 -mol(x,y) | in(x,y). [clausify(23)].
67 mol(x,y) | -in(x,y). [clausify(23)].
68 -mol(c1,c2). [deny(24)].
69 -mol(x,c1) | mol(x,c2) | -point(x). [resolve(27,a,25,a)].
70 line(c1). [resolve(30,a,29,a)].
71 plane(c2). [resolve(33,a,32,a)].
81 -in(c1,c2). [ur(67,a,68,a)].
97 -line(x) | in(x,c2) | in(f2(x,c2),x). [resolve(71,a,41,b)].
98 -line(x) | in(x,c2) | point(f2(x,c2)). [resolve(71,a,40,b)].
100 -in(f2(c1,c2),c2). [ur(42,a,70,a,b,71,a,c,81,a)].
275 in(f2(c1,c2),c1). [resolve(97,a,70,a),unit_del(a,81)].
300 point(f2(c1,c2)). [resolve(98,a,70,a),unit_del(a,81)].
305 -mol(f2(c1,c2),c2). [ur(66,b,100,a)].
336 -mol(f2(c1,c2),c1). [ur(69,b,305,a,c,300,a)].
```

```
390 $F. [ur(67,a,336,a),unit_del(a,275)].

============================== end of proof ==========================
```

Proof D.5.12: $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #12 in T_{most_group}

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3677 was started by cchui on MacBook-Pro.local,
Thu Nov 8 18:55:24 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 0.05 (+ 0.00) seconds.
% Length of proof is 15.
% Level of proof is 3.
% Maximum clause weight is 18.
% Given clauses 36.

2 (all q all x all y all z (planar_pendant(x,q) & planar_pendant(y,q) &
   ↪ planar_pendant(z,q) -> x = y | y = z | z = x)) # label(non_clause). [
   ↪ assumption].
22 (all x all y (end(x,y) <-> planar_pendant(x,y))) # label(non_clause). [
   ↪ assumption].
24 (all x all y all z all u (group(x) & end(y,x) & end(z,x) & end(u,x) -> y
   ↪  = z | y = u | z = u)) # label(non_clause) # label(goal). [goal].
33 -end(x,y) | planar_pendant(x,y). [clausify(22)].
34 end(c2,c1). [deny(24)].
35 end(c3,c1). [deny(24)].
36 end(c4,c1). [deny(24)].
40 -planar_pendant(x,y) | -planar_pendant(z,y) | -planar_pendant(u,y) | z =
   ↪  x | u = z | u = x. [clausify(2)].
69 c3 != c2. [deny(24)].
70 c4 != c2. [deny(24)].
71 c4 != c3. [deny(24)].
73 planar_pendant(c2,c1). [resolve(34,a,33,a)].
74 planar_pendant(c3,c1). [resolve(35,a,33,a)].
75 planar_pendant(c4,c1). [resolve(36,a,33,a)].
94 $F. [ur(40,b,73,a,c,74,a,d,70,a(flip),e,69,a,f,71,a(flip)),unit_del(a
   ↪ ,75)].

============================== end of proof ==========================
```

Proof D.5.13: $T_{cycle\_path\_subgraph\_nonisolated} \cup \Pi_2 \models$ Axiom #13 in T_{most_group}

```
============================== prooftrans ============================
Prover9 (32) version Dec-2007, Dec 2007.
Process 3684 was started by cchui on MacBook-Pro.local,
```

```
Thu Nov 8 18:55:39 2018
The command was "/Applications/Prover9-Mace4-v05B.app/Contents/Resources/
   ↪ bin-mac-intel/prover9".
============================== end of head ===========================

============================== end of input ==========================

============================== PROOF =================================

% -------- Comments from original proof --------
% Proof 1 at 2.47 (+ 0.01) seconds.
% Length of proof is 57.
% Level of proof is 3.
% Maximum clause weight is 64.
% Given clauses 53.

1 (all x all y all z all u all q all l1 all l2 all l3 (plane(q) & point(x)
   ↪ & point(y) & point(z) & point(u) & in(x,q) & in(y,q) & in(z,q) & in(u
   ↪ ,q) & line(l1) & in(l1,q) & in(x,l1) & in(y,l1) & line(l2) & in(l2,q)
   ↪  & in(x,l2) & in(z,l2) & line(l3) & in(l3,q) & in(x,l3) & in(u,l3) ->
   ↪  u = z | y = z | u = y)) # label(non_clause). [assumption].
19 (all x (atom(x) <-> point(x))) # label(non_clause). [assumption].
20 (all x (bond(x) <-> line(x))) # label(non_clause). [assumption].
21 (all x (group(x) <-> plane(x))) # label(non_clause). [assumption].
23 (all x all y (mol(x,y) <-> in(x,y))) # label(non_clause). [assumption].
24 (all a1 all a2 all a3 all a4 all g all b1 all b2 all b3 (group(g) & atom
   ↪ (a1) & atom(a2) & atom(a3) & atom(a4) & mol(a1,g) & mol(a2,g) & mol(
   ↪ a3,g) & mol(a4,g) & bond(b1) & mol(b1,g) & mol(a1,b1) & mol(a2,b1) &
   ↪ bond(b2) & mol(b2,g) & mol(a1,b2) & mol(a3,b2) & bond(b3) & mol(b3,g)
   ↪  & mol(a1,b3) & mol(a4,b3) -> a4 = a3 | a2 = a3 | a4 = a2)) # label(
   ↪ non_clause) # label(goal). [goal].
26 -atom(x) | point(x). [clausify(19)].
27 atom(c1). [deny(24)].
28 atom(c2). [deny(24)].
29 atom(c3). [deny(24)].
30 atom(c4). [deny(24)].
32 -bond(x) | line(x). [clausify(20)].
33 bond(c6). [deny(24)].
34 bond(c7). [deny(24)].
35 bond(c8). [deny(24)].
37 -group(x) | plane(x). [clausify(21)].
38 group(c5). [deny(24)].
42 -mol(x,y) | in(x,y). [clausify(23)].
43 mol(c1,c5). [deny(24)].
44 mol(c2,c5). [deny(24)].
45 mol(c3,c5). [deny(24)].
46 mol(c4,c5). [deny(24)].
47 mol(c6,c5). [deny(24)].
48 mol(c1,c6). [deny(24)].
49 mol(c2,c6). [deny(24)].
50 mol(c7,c5). [deny(24)].
51 mol(c1,c7). [deny(24)].
52 mol(c3,c7). [deny(24)].
53 mol(c8,c5). [deny(24)].
```

```
54 mol(c1,c8). [deny(24)].
55 mol(c4,c8). [deny(24)].
56 -plane(x) | -point(y) | -point(z) | -point(u) | -point(w) | -in(y,x) | -
   ↪ in(z,x) | -in(u,x) | -in(w,x) | -line(v5) | -in(v5,x) | -in(y,v5) | -
   ↪ in(z,v5) | -line(v6) | -in(v6,x) | -in(y,v6) | -in(u,v6) | -line(v7)
   ↪ | -in(v7,x) | -in(y,v7) | -in(w,v7) | w = u | u = z | w = z. [
   ↪ clausify(1)].
86 c4 != c3. [deny(24)].
87 c3 != c2. [deny(24)].
88 c4 != c2. [deny(24)].
89 point(c1). [resolve(27,a,26,a)].
90 point(c2). [resolve(28,a,26,a)].
91 point(c3). [resolve(29,a,26,a)].
92 point(c4). [resolve(30,a,26,a)].
93 line(c6). [resolve(33,a,32,a)].
94 line(c7). [resolve(34,a,32,a)].
95 line(c8). [resolve(35,a,32,a)].
96 plane(c5). [resolve(38,a,37,a)].
97 in(c1,c5). [resolve(43,a,42,a)].
98 in(c2,c5). [resolve(44,a,42,a)].
99 in(c3,c5). [resolve(45,a,42,a)].
100 in(c4,c5). [resolve(46,a,42,a)].
101 in(c6,c5). [resolve(47,a,42,a)].
102 in(c1,c6). [resolve(48,a,42,a)].
103 in(c2,c6). [resolve(49,a,42,a)].
104 in(c7,c5). [resolve(50,a,42,a)].
105 in(c1,c7). [resolve(51,a,42,a)].
106 in(c3,c7). [resolve(52,a,42,a)].
107 in(c8,c5). [resolve(53,a,42,a)].
108 in(c1,c8). [resolve(54,a,42,a)].
109 in(c4,c8). [resolve(55,a,42,a)].
448 $F. [ur(56,a,96,a,b,89,a,c,91,a,d,90,a,e,92,a,f,97,a,g,99,a,h,98,a,i
   ↪ ,100,a,j,94,a,k,104,a,l,105,a,m,106,a,n,93,a,o,101,a,p,102,a,q,103,a,
   ↪ r,95,a,s,107,a,t,108,a,v,88,a,w,87,a(flip),x,86,a),unit_del(a,109)].

============================== end of proof ==========================
```