Fall 12-2021

Improving Model Finding for Integrated Quantitative-qualitative Spatial Reasoning With First-order Logic Ontologies

Shirly Stephen
University of Maine, shirly.rock@gmail.com

IMPROVING MODEL FINDING FOR INTEGRATED QUANTITATIVE-QUALITATIVE SPATIAL REASONING WITH FIRST-ORDER LOGIC ONTOLOGIES

By

Shirly Stephen

B.S. - Anna University 2011

M.S. - University of Maine 2016

A DISSERTATION

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

(in Spatial Information Science and Engineering)

The Graduate School

The University of Maine

December 2021

Advisory Committee:

Dr. Torsten Hahmann, Associate Professor of Spatial Computing, Advisor

Dr. Kate Beard Tisdale, Professor of Spatial Computing

Dr. Roy Turner, Associate Professor of Computer Science

Dr. Silvia Nittel, Associate Professor of Spatial Computing

Dr. Sepideh Ghanavati, Assistant Prodessor of Computer Science

IMPROVING MODEL FINDING FOR INTEGRATED QUANTITATIVE-QUALITATIVE SPATIAL REASONING WITH FIRST-ORDER LOGIC ONTOLOGIES

By Shirly Stephen

Dissertation Advisor: Dr. Torsten Hahmann

An Abstract of the Dissertation Presented
in Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy
(in Spatial Information Science and Engineering)
December 2021

Many spatial standards are developed to harmonize the semantics and specifications of GIS data and for sophisticated reasoning. All these standards include some types of simple and complex geometric features, and some of them incorporate simple mereotopological relations. But the relations as used in these standards, only allow the extraction of qualitative information from geometric data and lack formal semantics that link geometric representations with mereotopological or other qualitative relations. This impedes integrated reasoning over qualitative data obtained from geometric sources and "native" topological information – for example as provided from textual sources where precise locations or spatial extents are unknown or unknowable. To address this issue, the first contribution in this dissertation is a first-order logical ontology that treats geometric features (e.g. polylines, polygons) and relations between them as specializations of more general types of features (e.g. any kind of 2D or 1D features) and mereotopological relations between them. Key to this endeavor is the use of a *multidimensional* theory of space wherein, unlike traditional logical theories of mereotopology (like RCC), spatial entities of different dimensions can co-exist and be related.

However terminating or tractable reasoning with such an expressive ontology and potentially large amounts of data is a challenging AI problem. Model finding tools used to verify FOL

ontologies with data usually employ a SAT solver to determine the satisfiability of the propositional instantiations (SAT problems) of the ontology. These solvers often experience scalability issues with increasing number of objects and size and complexity of the ontology, limiting its use to ontologies with small signatures and building small models with less than 20 objects. To investigate how an ontology influences the size of its SAT translation and consequently the model finder's performance, we develop a formalization of FOL ontologies with data. We theoretically identify parameters of an ontology that significantly contribute to the dramatic growth in size of the SAT problem. The search space of the SAT problem is exponential in the signature of the ontology (the number of predicates in the axiomatization and any additional predicates from skolemization) and the number of distinct objects in the model. Axiomatizations that contain many definitions lead to large number of SAT propositional clauses. This is from the conversion of biconditionals to clausal form. We therefore postulate that optional definitions are ideal sentences that can be eliminated from an ontology to boost model finder's performance. We then formalize optional definition elimination (ODE) as an FOL ontology preprocessing step and test the simplification on a set of spatial benchmark problems to generate smaller SAT problems (with fewer clauses and variables) without changing the satisfiability and semantic meaning of the problem. We experimentally demonstrate that the reduction in SAT problem size also leads to improved model finding with state-of-the-art model finders, with speedups of 10-99%. Altogether, this dissertation improves spatial reasoning capabilities using FOL ontologies – in terms of a formal framework for integrated qualitative-geometric reasoning, and specific ontology preprocessing steps that can be built into automated reasoners to achieve better speedups in model finding times, and scalability with moderately-sized datasets.

DEDICATION

Dedicated to my parents who always encouraged me to never stop learning, growing, and adapting.

ACKNOWLEDGEMENTS

This dissertation is the culmination of the support, encouragement, and guidance from a number of people. I would like to thank them all.

Firstly, I would like to thank my academic advisor, Dr. Torsten Hahmann, for his mentoring, continuous support, constant encouragement, and whose expertise was invaluable in bringing this dissertation to fruition. His never-ending patience to explain concepts and clarify all the little details during the countless discussions helped me in the completion of this work. I am also thankful to him for carefully reviewing and commenting on several versions of this thesis.

I would like to acknowledge my deep sense of gratitude to the members on my thesis committee (Dr. Kate-Beard Tisdale, Dr. Roy Turner, Dr. Silvia Nittel, and Dr. Sepideh Ghanavati) for their support throughout my graduate career. Further, I would like to extend my thanks to the National Science Foundation for aiding my research financially.

I thank all my friends and colleagues at the SIE department and at the University of Maine for all the good times that made my graduate life a memorable one. In particular, special thanks to Sabrina, Radowan, Sudheera, Phani, Siri, and Salomi. I am at lost for words when expressing my gratitude to my beloved parents and siblings for their constant encouragement, blessings, and cheering for me in all my academic and personal endeavours.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ATP	Automated Theorem Prover
BCE	Blocked Clause Elimination
BCP	Boolean Constraint Propagation
CADE	Conference on Automated Deduction
CASC	CADE ATP System Competition
CDCL	Conflict-Drivern Clause Learning
CNF	Clausal Normal Form or Conjunctive Normal Form
CODI	COntainment-Dimension ontology
CODIB	COntainment-Dimension-Boundary ontology
COLORE	COmmon Logic Ontology REpository
DPLL	Davis-Putnam-Loveland-Longland procedure
EPR	Efficient Propositional fragment
FOL	First-Order Logic
FOF	First Order Form
GIS	Geographic Information Systems
ODE	Optional Definition Elimination
PPE	Pure Predicate Elimination
QBF	Quantitative Boolean Formula
QSR	Qualitative Spatial Reasoning
RCC	Region Connection Calculus
SAT	SATisfiability problem
SFA	Simple Feature Access model
TPTP	Thousands of Problems for Theorem Proving
UDE	Unused Definition Elimination
9-IM	9-Intersection Matrix

CHAPTER 1

INTRODUCTION

1.1 Context and Motivation

Big data has revolutionized informed decision-making by allowing the extraction of valuable insights and opportunities from a range of reliable data. The explosive growth of geospatial data has made it a valuable commodity as there are major markets for it and new opportunities (such as drone technology and unmanned vehicles) unfolding every day. This has driven the development of many industrial and generic spatial standards to enable the effective reuse of data and for the sharing of information and interoperability across applications. Axiomatic representations enable reasoning consistent with common-sense reasoning [94] in varying degrees. Although such standards and spatial representations cover a broad range of data types, there are emerging spatial data reasoning tasks that are not entirely supported. It concerns cases that require identifying locations from a set of qualitative spatial constraints and qualitative information obtained from geometric and non-geometric sources (e.g. a live twitter feed), as the vast majority of digital spatial data is available as both geometric shapefiles and text. From a certain perspective, the input consists of a set of formal qualitative assertions and a set of qualitative statements abstracted from geometric datasets, and the expected output is either an entity's name, or relationship between a set of entities. Such requests that require integrated qualitative-geometric spatial reasoning, like the example we describe below can occur in everyday life demands.

Example: Road segment search - We have an incident report about a broken gas pipeline in the street between *St.Johns hospital* and *Thomas HS* from a live twitter feed 'Gas pipe broken between St.Johns and Thomas school #BangorGas'. We are looking for this specific segment of a larger road that needs to be closed off to public due to this disaster. We may not know the name or precise address, but we know a few spatial constraints about where the road segment may be. Qualitative information about this road obtained from the twitter report indicates it abuts *St.Johns hospital* and *Thomas HS*. In order to shut down the power supply on that street, the control center at the utility management company needs to identify the exact road segment in question, as the tweet does not mention the road(s) but only *implicitly* refers to it. This information can easily be extracted from freely and readily available geometric base maps. Figure 1.1 depicts the part of the OpenStreetMap dataset relevant for the entities of interest.

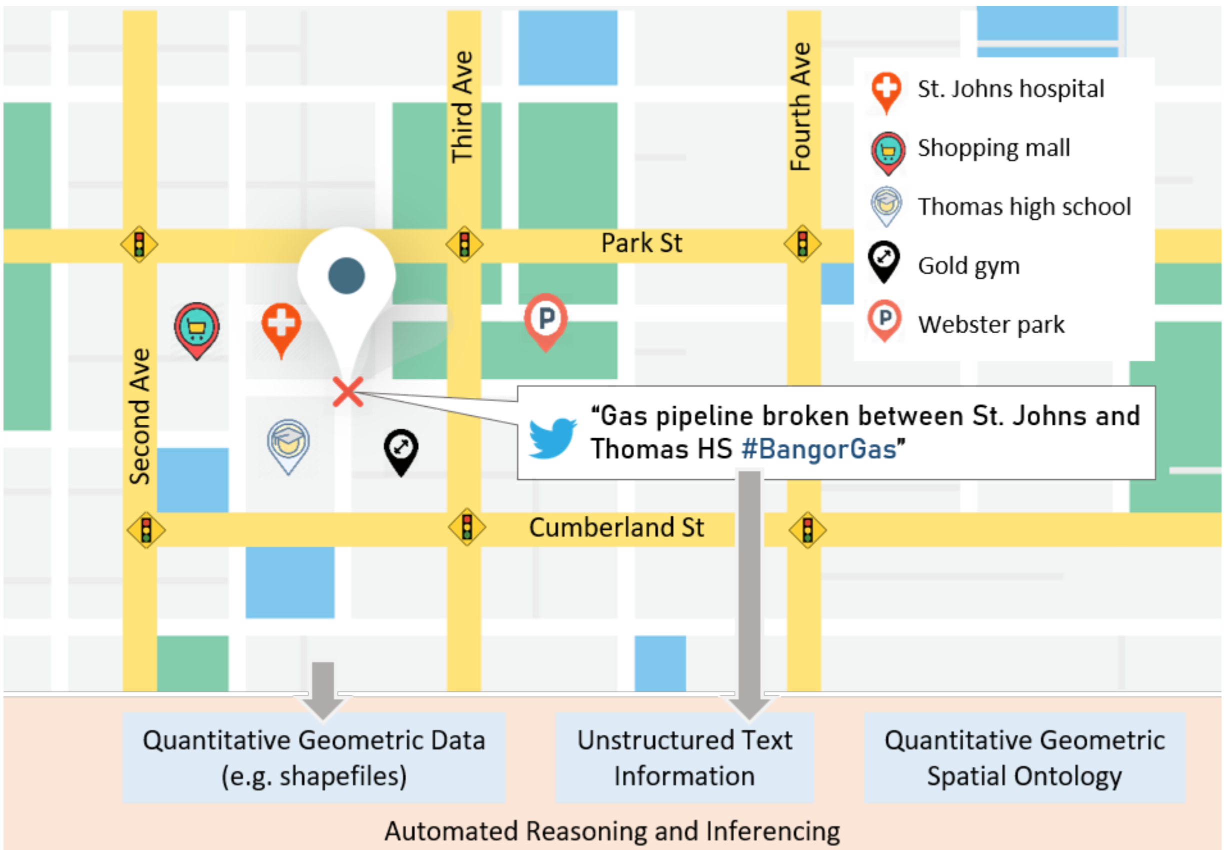


Figure 1.1: Section of OpenStreetMap dataset relevant for the entities of interest. The map highlights all the relevant features of interest mentioned in the tweet, and connected spatial entities whose qualitative information is needed for the query.

1.2 Objectives

The scenario described above involves integrated reasoning over mixed spatial data: qualitative relations between non-geometric entities and geometric objects and their relations using first-order logic (FOL) ontologies – specifically through model finding. The overarching objective of this dissertation is "*to demonstrate that model finding over FOL ontologies of qualitative space with small geometric datasets is feasible and can be used to externally verify these ontologies*". Broadly, the entire dissertation focuses on two details,

- the *representational* aspect that requires the development of an integrated framework of qualitative and geometric concepts, and
- the *reasoning* aspect to test the feasibility of tractable reasoning using the spatial ontology with medium-sized datasets in FOL.

1.2.1 Challenges

While trying to accomplish the underlying objective of this dissertation, which is to enable joint qualitative reasoning over geometric and qualitative spatial information and to demonstrate the feasibility of reasoning over medium-sized spatial datasets the following two challenges arise.

1. Existing algorithms cannot extract meaningful information that combine geometric and natural language based qualitative spatial descriptions, let alone reason and query with the combined knowledge. The Simple Feature Access (SFA) model [151] is an OGC/ISO standard that standardizes spatial operations and simple topological and mereotopological relations over geometric features such as points, line segments, polylines, polygons, and polyhedral surfaces. The SFA standard is of specific interest for the following reasons: (1) it is implemented in common spatial databases such as ESRI ArcGIS and PostGIS for accessing and storing spatial data, and it also forms the vector data basis for libraries such as GDAL and the GeoJSON standard, (2) it is a widely used data interchange standard used by many other OGC/ISO standards such as GeoSPARQL [220] and Observation and Measurements [69], (3) its relations are based on the well-studied and commonly accepted 9-intersection relations and the RCC relations. But while SFA's supplied relations enable

qualitative querying over the geometric features, the relations' semantics are not formalized and therefore have weak precision. The lack of formalization prevents further automated reasoning – apart from simple querying – with the geometric data, either in isolation or in conjunction with external purely qualitative information as one might extract from textual sources, such as social media. Summarily, current specifications and standards do not allow pure qualitative reasoning through the abstraction[1] of qualitative information from geometric and non-geometric data sources. To realize the kind of integration of qualitative and geometric information described in the example in Figure 1.1, a formal spatial representation that combines geometric and qualitative concepts, which will allow integrated reasoning using FOL reasoning engines is needed in hand.

2. The main purpose of constructing a formal ontology for integrated spatial reasoning is to enable efficient decision-making when combined with real-world domain data. However, despite remarkable advances in the development of decision procedures and reasoning engines, achieving terminating or tractable reasoning in FOL with large datasets remains a challenging problem. Model finders, the class of automated tools to verify FOL ontologies against datasets, traditionally translate the problem into an equivalent propositional satisfiability (SAT) problem and then tackle it using a propositional SAT solver. Although SAT is an NP-hard problem and thus generally intractable, through efficient heuristics and simplifications many instances of propositional problems are easily solved in practice. Unfortunately, SAT solvers often experience scalability issues when trying to construct models for FOL ontologies in conjunction with even moderately sized datasets as the size of the SAT problem exponentially increases with increasing number of objects and size and complexity of the ontology (i.e. the axiomatization). There any no works currently that clearly identify the key parameters that contribute to the exponential explosion of a SAT problem when translated from FOL. When switching from (geometric) objects to

[1]This refers to spatial metric and coordinate information abstraction. For example, preserving the notion that a road is a curve but without any additional numeric information.

(qualitative spatial) relations describing them, the amount of data to be considered is subject to a combinatorial explosion. Let us assume, for simplicity, we are only interested in contact $C(x, y)$, then every pair of objects in a spatial dataset raises one qualitative relation (positive or negated). Current model finding works for FOL ontologies has typically been limited to small models with less than 20 objects, and available model finders, such as Paradox [56] or Mace4 [200], have mostly been tested on relatively small axiomatizations with very small signatures. Accordingly, one of the main challenges in FOL reasoning is that of tractable reasoning with a complex spatial ontology and potentially large amounts of data points (i.e. assertions).

1.2.2 Specific Objectives

Toward the overarching objective and the associated challenges, the dissertation specifically addresses the following objectives, which will be discussed summarily in the next section:

O1. Explicitly formalize the semantics between qualitative and geometric spatial representations to enable spatial reasoning and querying (1) of a mix of qualitative and geometric data, (2) about purely qualitative information over geometric data.

O2. Develop a formal framework for size or complexity measures of ontologies with data for FOL reasoning.

O3. Identify specific size measures that have the greatest impact on the hardness of FOL model finding.

O4. Develop and evaluate a simplification method to limit the growth of the satisfiability (SAT) search space for FOL model finding problems.

1.3 Contributions

The two main contributions made in this dissertation are (1) laying the representational foundation that enables integrated qualitative reasoning over geometric and qualitative spatial information thereby addressing O1 - Section 1.3.1; (2) analyzing how the terminology used within an ontology influences model finder performance and present tests that depict how the elimination of optional definitions help model finding to scale better in practice, thereby addressing O2-O4 - Section 1.3.2.

1.3.1 Spatial Representation for Integrated Reasoning

We develop and encode an integrated semantics of spatial information – geometric configurations and qualitative spatial relations – that reuses concepts from, but is also schematically distinct from existing axiomatic representations and spatial data standards. Specifically, we formalize the semantics of SFA's geometric features and mereotopological relations, called the SF-FOL, by defining or restricting them in terms of the spatial entity types and relations provided by CODIB [128], a first-order logical theory from an existing logical formalization of multidimensional qualitative space. The resulting spatial ontology (in Chapter 4) allows using geometric and qualitative information for pure qualitative spatial reasoning as well as mixed geometric-qualitative reasoning cases as illustrated in the example in Section 1.1. The ontology is formalized in first-order logic, which allows reasoning using first-order logic reasoners. Although this work specifically aims to enable reasoning over a mix of information from geometric datasets and qualitative sources such as natural language spatial text, we anticipate wider applications of the ontology. It can serve as a formal spatial interoperability standard for FOL ontologies of spatial relations such as RCC [1] and INCH [2], but also domain ontologies such as the GWML2 [133] and the National Map [271].

1.3.2 Model Finding for Spatial Ontologies

Because FOL is a very rich representation language, and computational reasoning with anS axiomatization becomes quickly intractable in practice, and in the presence of data is believed to be entirely infeasible, spatial reasoning with formal ontologies using model finders is a task that is never undertaken. In the second part of the dissertation we make the following specific contributions:

1. We develop a formalization of FOL ontologies with data, and present a study of various measures that contribute to the size of the resulting SAT problems (Chapter 6).

2. We introduce *optional definition elimination* (ODE) as a preprocessing technique applied to an FOL ontology and investigate its impact in generating smaller SAT problems (with fewer clauses and variables) without changing the satisfiability and semantic meaning of the problem (Chapter 5).

3. We implement ODE simplification on a set of spatial benchmark problems and conduct a twofold study. First in Chapter 6, we show a theoretical calculation of size measures based on the terminology of an ontology and the number of distinct objects described in the data. Then through the experimental study we demonstrate how these measures correlate to the size of the resulting SAT problem, which determines the size of the search space for model finders in Chapter 7.

Results are reported from experiments with the benchmark problems using three state-of-the -art model finders: Paradox, Vampire and iProver. We found that with ODE we were able to solve problems that were previously intractable, and model finding times with the best model finders decreased on average by 10% and sometimes up to 99%. The theoretical and experimental developments presented in this dissertation can be used to implement specific preprocessing steps that can be built into model finding tools. This will provide a small step towards enhancing reasoning capabilities – in terms of better speedups in model finding times, and scalability with data objects – a (extensively axiomatized) complex ontology such as SF-FOL against moderately sized (spatial) real-world data.

1.4 Overview

The rest of the dissertation is organized as follows:

- **Chapter 2** briefly reviews the syntax and semantics of first-order logic and propositional logic. The most popular SAT procedure – the DPLL algorithm – is introduced along with details of its modern implementation strategies. We introduce FOL model finding via SAT solving and the key steps involved in converting an FOL ontology into a propositional SAT problem. We then briefly review some ontologies of qualitative and geometric space with specific focus on CODI, RCC, and INCH calculus, which are used as benchmark ontologies for studying the scalability of model finding and potential improvements.

- **Chapter 3** reviews some previous work upon which our research draws or that is related in aim or methodology, and highlights their differences from this dissertation's work and their limitations. The related work includes work corresponding to the development of formal qualitative spatial formalisms, and work related to SAT-based FOL model finding.

- **Chapter 4** presents the formalization of the Simple Feature Access spatial concepts and relations as an extension of CODI and CODIB in first-order logic. This chapter was published in [256].

- **Chapter 5** presents a formalization of the concepts of TBox, ABox and sets of removable definitions for FOL ontologies. It studies how different measures of an FOL ontology influence the size of the corresponding SAT problem. Then ODE is introduced as an FOL preprocessing technique to dramatically reduce the size of the resulting SAT problem and thereby to alleviate some difficulty during model finding.

- **Chapter 6** analyzes the optional definition elimination technique developed in Chapter 5, with respect to how it reduces key size attributes - especially the number of propositional variables - in the resulting SAT problems and what side affects it has on other measures (e.g. number of clauses, the length or complexity of clauses, etc.) on spatial benchmark problems with different sized datasets.

- **Chapter 7** experimentally analyses the performance of three model finders on the set of spatial problems constructed in Chapter 6 with different degrees of ODE performed and compares it to the runtimes without ODE. It then studies how the runtimes correlate to the calculated size attributes in order to identify which size attribute may be used as indicator to predict runtime via an automated preprocessing step.

- **Chapter 8** summarizes the main ideas of this dissertation, and suggests directions for future work.

CHAPTER 2

PRELIMINARIES

In this chapter, we will introduce the basic concepts of first-order logic as the language in which the ontologies in this dissertation are represented and first-order logic ontology verification that is fundamental to this dissertation. We will also overview the three ontologies of qualitative spatial relations that are used as benchmarks for the studies conducted in Chapters 6 and 7.

2.1 First-Order Logic Ontologies

First-order logic (FOL) also called predicate logic is widely used in formalizing semantics of domain, application, and upper ontologies [256, 133, 49, 123, 124], mathematical theories [248, 42], software and hardware verification tasks [55, 217, 171, 242]. These formal axiomatizations provide the background knowledge necessary to (1) prove conjectures, or in a computational sense for query answering tasks, (2) interpret a dataset in the domain, (3) semantically integrate different datasets or applications, or (4) make implicit assumptions in the domain explicitly provable for decision-making. The definitions and notations of FOL mentioned here are quite standard and mostly adopted from [29].

2.1.1 Syntax of First-Order Logic

An FOL ontology $\mathcal{O}$ is a set of FOL sentences σ using a particular language. The non-logical symbols, i.e. all constants, function symbols, and predicates, mentioned in O form its *vocabulary* or *signature*, denoted by $\lambda(\mathcal{O})$ (cf. Def.1). For simplicity, we consider here only ontologies with predicates and constants in their signatures, because each n-ary function symbol can be encoded as a n+1-ary predicate symbol by adding axioms that capture its functional nature[1] .

[1]Because constants typically represent objects from the domain of interest, we include them to allow specifying factual knowledge, i.e. data points.

Definition 1. *The* ***signature*** *of an ontology* $\mathcal{O}$, $\lambda(\mathcal{O})$ *is a tuple* $\sigma = (\mathbb{P}, a)$, *where* $\mathbb{P}$ *is an enumerable set of predicate symbols (or operators) and* $a : \mathbb{P} \rightarrow N$ *is a function describing the arity of the predicate symbols, with each predicate* $\Omega \in \mathbb{P}$ *having the arity* $a(\Omega) \geq 0$, *and constants have arity 0.*

Sentences are built up recursively from terms, atoms (FOL literals), and formulae.

Definition 2. *A* ***term*** *is simply an expression of the form* $\Omega(t_1, ..., t_n)$ *where* Ω *is a predicate symbol described by a signature* $\lambda(\mathcal{O})$ *of arity* a *and all* t_i *are atoms.*

Since we restrict ourselves to function-free signatures, *atoms* are either constants or variables.

Definition 3. *A* ***FOL literal*** *(often also called an atom) is a term or its negation* $\neg\Omega(t_1, ..., t_n)$.

An FOL *formula* in $\mathcal{O}$ is constructed from $\mathcal{L}$-atoms (or literals) using the logical connectives $\wedge, \vee, \rightarrow, \leftrightarrow$ and $\neg$ and/or the quantifiers $\forall$ and $\exists$ over FOL variables. Such a formula F is recursively constructed according to the following grammar:

$F ::= \Omega(t_1, ..., t_n) \mid \top \mid \bot \mid (\neg F) \mid (F_1 \wedge F_2) \mid (F_1 \vee F_2) \mid (F_1 \rightarrow F_2) \mid (F_1 \leftrightarrow F_2) \mid$
$(\forall v : F) \mid (\exists v : F)$

An FOL *sentence* is a closed formula wherein no variables appear free, i.e. all variables are within the scope of quantifiers. In the ontological sense, there are two primary types of sentences: terminological sentences, which constitute the TBox, and the assertional sentences, which form the ABox. Here we present a basic definition for an FOL ontology, and provide a more accurate formalization of the TBox and ABox in Chapter 5.

Definition 4. *An FOL ontology* $\mathcal{O}$ *is a set of FOL sentences (axioms and definitions) in a language* $\mathcal{L}(\mathcal{O})$ *that only use non-logical symbols from* $\lambda(\mathcal{O})$.

A formula is *ground* if there are no occurrences of variables – free or bound, i.e. with constants as the only terms. In a first-order specification, these terms typically represent objects from the domain that we want to reason about. A *theory* also called an *ontology* is any set of closed formulae.

2.1.2 Semantics of First-Order Logic

The semantics describe the meaning of, or how truth values are assigned to FOL formulae. Each FOL ontology $\mathcal{O}$ admits a set of *interpretations* as defined in Def. 5 from [127] over a nonempty domain D of individuals.

Definition 5. *An **interpretation** of an ontology $\mathcal{O}$ is a tuple $\mathcal{I} = \langle D, \Phi, \Psi \rangle$ that assigns a meaning to every symbol in the signature $\lambda(\mathcal{O})$. D denotes a nonempty domain, Φ a mapping of each variable in $\lambda(\mathcal{O})$ to an individual in D, Ψ is a mapping of all n-ary predicates $\Omega \in \lambda(\mathcal{O})$ to relations $\Psi(\Omega) : \mathcal{O}^n \rightarrow \{\texttt{True}, \texttt{False}\}$ where `True` means the relation holds and `False` means the relation does not hold.*

An interpretation $\mathcal{I}$ for which all sentences in $\mathcal{O}$ are true (i.e. all sentences are *satisfied* in $\mathcal{O}$) is called a *model* M, we write $M \models \mathcal{O}$ iff $M \models \psi$ for every $\psi \in \mathcal{O}$. An ontology is *consistent* (or *satisfiable*) if it has some model.

Definition 6. *An FOL sentence σ that uses only the nonlogical symbols from $\lambda(\mathcal{O})$ and that is true in every model of $\mathcal{O}$ is called a **theorem** of $\mathcal{O}$, written as $\mathcal{O} \models \sigma$. We then say the ontology $\mathcal{O}$ logically implies, or **entails** such a sentence σ.*

Because of the undecidability of FOL, we can eventually prove an ontology to be unsatisfiable/inconsistent if it is so (i.e. a sentence that is `False` can be eventually proven to be entailed), but we may never be able to prove that a satisfiable/consistent ontology is so (i.e. a sentence that is `False` may never be disproved).

2.2 FOL Model Finding via Propositional SAT Solving

The propositional satisfiability problem (SAT) is the following: *Given a propositional formula F, does F have a satisfying assignment? And if there exists one find the actual satisfying assignment (model).* The SAT problem tries to determine that each clause should have at least one literal that is true under the assignment in order to be satisfied. If there is no assignment satisfying all clauses, the formula is said to be unsatisfiable. The tools to answer this question are called satisfiability or SAT solvers, most of which which require the input propositional formula in Conjunction Normal Form (CNF).

In this section we first review the syntax and semantics of propositional logic, the language that is used to represent model finding instances for FOL ontology verification. We then describe the basic SAT algorithm and popular SAT solver techniques for propositional formula verification.

2.2.1 Syntax and Semantics of Propositional Logic

A *propositional literal* is a propositional (or boolean) variable v or its negation $\neg v$ that takes value in the set $\{\texttt{True,False}\}$. A *propositional formula* F is a logic expression defined over variables using boolean operators ($\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$) using the following grammar:

$F ::= v \mid (\neg F) \mid (F_1 \wedge F_2) \mid (F_1 \vee F_2) \mid (F_1 \rightarrow F_2) \mid (F_1 \leftrightarrow F_2)$.

A *propositional clause* is a disjunction of a set of literals to state propositions, and a conjunction of clauses form the formula F. A clause that contains only positive literals is called a *positive clause*. Similarly, a clause that contains only negative literals is a *negative clause*. A clause that contains at most one positive literal is called *Horn*. An *assignment* (similar to an interpretation in FOL) for a formula F is a mapping from literals to truth values $\sigma : V \rightarrow \{ \texttt{True,False} \}$. A *satisfying assignment* (i.e. similar to a model in FOL) for F is an assignment σ such that F evaluates to `TRUE` under σ. Accordingly, F is satisfiable if there exists a propositional assignment that satisfies F under the usual semantics for the logical connectives.

2.2.2 Model Finding via Translation to CNF and SAT

To facilitate automated reasoning, including model finding, an FOL ontology is typically converted to an equisatisfiable clausal normal form (which we call the FOL-CNF representation and formalized as $\mathcal{O}_{\text{FOL-CNF}}$ in Chapter 5) through the process of clausification. A formula is in clause normal form or *Conjunction Normal Form* (CNF) if it is a conjunction of clauses (cf. Def. 7), where variables in the clause may be universally quantified.

Definition 7. *A **FOL clause** is a disjunction of literals $L_1 \vee ... \vee L_n$, where $n \geq 0$. When $n = 0$, it is the **empty clause**, whereas if the clause contains a single literal, i.e. $n = 1$, it is called a **unit clause**.*

Finding a model of the FOL ontology can then be achieved by showing satisfiability of its equivalent FOL-CNF problem through propositionalization. A detailed description of this two-staged process is presented in Chapter 5, but we describe clausification in detail here.

Clausification - First-Order Formula Transformation to CNF. A formula in FOL is translated to FOL-CNF through a 7-step process adopted from the Skolem's algorithm [29]. This is illustrated here using the FOL definition for contact σ_C from the CODI ontology [128] as an example:

$$(\sigma_C)\ \forall x, y\ C(x, y) \leftrightarrow \exists z\, [Cont(z, x) \wedge Cont(z, y)] \qquad \textbf{(CODI contact)}$$

1. Standardize variables by renaming bound variables to ensure each quantifier uses a unique variable. Unique variables are bound to quantifiers by default in σ_C.

2. Use logical equivalences to eliminate biconditionals and conditionals. First replace all biconditionals $\leftrightarrow$ by a conjunction of two implications – (a). Then replace implications by logically equivalent disjunctions – (b).

   (a) $\Big[\forall x, y\ C(x, y) \rightarrow \exists z\Big(Cont(z, x) \wedge Cont(z, y)\Big)\Big] \wedge \Big[\exists z\Big(Cont(z, x) \wedge Cont(z, y)\Big) \rightarrow \forall x, y\ C(x, y)\Big]$

   (b) $\Big[\neg\forall x, y\ C(x, y) \vee \exists z\Big(Cont(z, x) \wedge Cont(z, y)\Big)\Big] \wedge \Big[\neg\exists z\Big(Cont(z, x) \vee Cont(z, y)\Big) \vee \forall x, y\ C(x, y)\Big]$

3. Move $\neg$ (if any) inwards using de Morgans's rule and simplify by moving all quantifiers outside of negations.

   $\Big[\exists x, y\ \neg C(x, y) \wedge \forall z\Big(\neg Cont(z, x) \vee \neg Cont(z, y)\Big)\Big] \wedge \Big[\forall z\Big(\neg Cont(z, x) \wedge \neg Cont(z, y)\Big) \wedge \exists x, y\ \neg C(x, y)\Big]$ – from the translation of **(b)**

4. Extract all quantifiers to the prefix of the sentence.

   $\exists x \exists y \forall z \; \neg C(x,y) \wedge \Big[\neg Cont(z,x) \vee \neg Cont(z,y)\Big] \wedge \Big[(\neg Cont(z,x) \wedge \neg Cont(z,y)) \wedge \neg C(x,y)\Big]$

5. Skolemization (cf. Def. 8) replaces each existential variable with a Skolem function. The arity of the function depends on the number of quantified variables within which the eliminated quantifier is nested.

   $\forall x, y, z \Big[\neg C(x,y) \vee Cont(f_1(x,y), x)\Big] \wedge \Big[\neg C(x,y) \vee Cont(f_1(x,y), y)\Big] \wedge$
   $\Big[C(x,y) \vee \neg Cont(z,x) \vee \neg Cont(z,y)\Big]$

   Definition 8. ***Skolemization*** *of a sentence σ replaces every existentially quantified variable $\exists x$ that is preceded with a set of universally quantified variables $y_1, \ldots, y_n$ by a new n-ary function symbol, called the Skolem function. If there are no universal quantifiers preceding $\exists x$, then x is replaced by a new constant (0-ary function)* [29].

6. Universal quantifiers are dropped and all unbound variables in the formula are now implicitly taken to be universally quantified.

   $\Big[\neg C(x,y) \vee Cont(f_1(x,y), x)\Big] \wedge \Big[\neg C(x,y) \vee Cont(f_1(x,y), y)\Big] \wedge \Big[C(x,y) \vee \neg Cont(z,x) \vee$
   $\neg Cont(z,y)\Big]$

7. Apply distributive law for conjunctions and disjunctions and simplify the formula.

   $\Big[\neg C(x,y) \vee Cont(f_1(x,y), x)\Big] \wedge \Big[\neg C(x,y) \vee Cont(f_1(x,y), y)\Big] \wedge \Big[C(x,y) \vee \neg Cont(z,x) \vee$
   $\neg Cont(z,y)\Big]$ – this is now an FOL-CNF formula with 3 clauses.

Conversion of FOL sentences to an FOL-CNF formula can lead to an exponential growth in length (via the distributive rule in step 7) of the formula and may introduce functions via skolemization of existential quantifiers. For example, if the original formula has $(2 \cdot n)$ literals, the corresponding CNF can have upto 2^n disjunctive clauses, each with n literals[2] .

[2]Definitional CNF's are alternative conversions to CNF that avoid this exponential growth. It introduces a new proposition variable R_i for each conjunctive clause $(P_i \wedge Q_i)$. Then if $M \models R_i$, then $M_j \models P_i$ and $M_j \models Q_i$. The resultant FOL-CNF is not significantly bigger than the original formula, but has more propositional variables). However we use the regular CNF-conversion method to determine clause count in FOL-CNF.

The FOL-CNF formula is then converted to a propositional SAT problem by instantiating the formula with elements from a domain set D. Each FOL variable x, y, and z assumes objects from $D = \{d_1, d_2, d_3, ..., d_n\}$. The formula from step [7] when instantiated for ($x = d_1$, $y = d_2$, and $z = d_3$) results in the following propositional formula: $(p_1 \vee p_2) \wedge (p_1 \vee p_3) \wedge (\neg p_1 \vee p_4 \vee p_5)$. Each grounded literal in the FOL-CNF formula now corresponds to a unique variable in the SAT problem called a propositional variable $(p_1, p_2, .., p_n)$, which assumes truth values from the set { `True,False` }. The FOL-CNF formula in [7] contains two binary predicates, which when instantiated for a domain D of size d results in $d^2 \cdot d^2$ propositional variables with a search space of 2^{2d^2} (i.e. when d = 10, #propositional variables = 10,000 and search space = 2^{200}). Thus in FOL there is combinatorial explosion of the search space based on the domain size and the number of predicates.

2.2.3 Decision Procedures for Determining Satisfiability

SAT is a classic NP-complete problem [66], meaning there is no known deterministic polynomial-time algorithm that can solve an arbitrary problem instance. The worst case scenario for deciding SAT involves trying all 2^n possible assignments for a formula with n variables. Best current complete methods are polynomial (indeed linear time) for 2-CNF and exponential for 3-CNF (SAT instances where all clauses have length 2 and 3 respectively). The practical importance of SAT in the fields of automated reasoning and artificial intelligence have led to the development of efficient decision procedures and algorithms that have been implemented into SAT solvers. It is also common for first-order logic problems to be reduced to propositional logic to determine their satisfiability using these solvers. In fact, we will employ FOL model finders that do exactly these as described in more detail in Section 2.3. As a consequence of a deeper understanding of sources of intractability, control measures to avoid exponential growth in problem size, and the availability of more powerful computing resources, it has been possible to develop solvers that handle industrial problems with millions of variables and constraints[3] as discussed in more detail in Section 2.2.5.

The literature distinguishes between two categories of decision procedures for satisfiability checking:

- A **complete** decision procedure is one that takes an input formula and always finds a solution (whether satisfiable or unsatisfiable), if it exists, in finite time. The first such satisfiability algorithm proposed by Davis and Putnam in 1960 [75], and later improved by Davis, Logemann, and Loveland (DLL) [74] is still the basic foundation of many modern SAT solvers. Since complete methods aim at exploring the entire solution space, this exhaustive search is too costly. Pruning techniques are therefore implemented to rapidly determine and ignore regions that contain no solution, and simplify formula size (we will discuss some of these simplification techniques in Section 3.2.2).

- An **incomplete** procedure is one that returns a solution when one is found, or returns 'unknown', when the search has run long enough without finding any solution. Such procedures are usually based on stochastic local search methods [148, 147] that start with an arbitrary truth assignment, make small changes to this assignment trying to get closer to a solution by heuristics without exhaustively exploring the search space. These algorithms are unable to determine the unsatisfiability of a formula. They are more efficient than complete ones, however there is not a lot of work using them to solve industrial problems. Several variants of the WalkSat algorithm [245] are some of the most successful implementations of local search.

The semantics of propositional logic satisfiability can be defined in terms of logical calculi and inference rules. Many inference systems have been defined for propositional logic

[3]However we remind the reader that this is the case for problems originating from propositional logic and not FOL, where solvers are mostly intractable with moderately large and complex problems. This is discussed in the context of related work in Chapter 3

(e.g.[204]), but the resolution rule is the most popular proof procedure (used in the DPLL algorithm described in the next section) and is defined as follows:

Definition 9. ***Resolution:*** *If two arbitrary clauses A and B have exactly one pair of complementary literals $a \in A$ and $\neg a \in B$, then the clause $A \vee B$ is called the resolvent (or consequence) of A and B.*

$$\frac{(A \vee a)(\neg a \vee B)}{A \vee B}$$

The resolvent can be added to the formula without changing its satisfiability.

2.2.4 The Davis Putnam Logemann Loveland (DPLL) Algorithm

The Davis-Putnam-Logemann-Loveland or **DPLL** procedure [75] is a classic complete SAT procedure that is still employed in modern SAT-solvers. DPLL is a later refinement of the original Davis and Putnam (DP) algorithm [75], which used the resolution rule (Def. 9). Most current complete SAT solvers extend the classic DPLL with three main features: branching, unit propagation[4] , and backtracking. In addition they incorporate many optimization strategies such as branching heuristics for variable selection, functions for clause learning, conflict analysis for pruning the search space, watched literals for efficient constraint propagation and backjumping, all to overcome the exponential build-up of clauses and search space that led to a very slow run time performance in original DP and DPLL procedures. In addition, several preprocessing steps are performed to simplify the problem before branching and to determine if the problem can be trivially satisfied before branching. These state-of-the-art algorithms are called conflict-driven clause learning (CDCL) algorithms, and is discussed in the upcoming section. Also note that DPLL requires the input as CNF formulae.

2.2.5 Improvements to DPLL

Over the past couple decades numerous improvements have been made to the DPLL algorithm by combining techniques such as good decision heuristics, simplification, compact

[4]Or Boolean Constraint Propagation (BCP) is the process of using partial assignments in order to iteratively fix (or assign) appropriate values to literals for a satisfying assignment for the formula.

data structures and conflict-driven learning techniques. This has led to the rise of SAT-Solvers (such as CHAFF [209], MINISAT [90]) that can solve instances with thousands and even millions of variables [43, 149], which make the use of SAT-solvers for verification of FOL ontologies as studied in this dissertation possible at all. Here, we will discuss some popular algorithmic improvements, and preprocessing techniques that simplify formula encodings is reviewed in Section 3.2.2.

Conflict Analysis and Backtracking: The backtracking search algorithm starts from an empty truth assignment and traverses the space of all truth assignments by maintaining a decision tree. Each node in the decision tree specifies an assignment of a Boolean value (true or false) to a variable. The search process extends the current assignment either by making an assignment to an unassigned variable or by making assignments following the logical consequences of the assignments made thus far. This deduction process may sometimes lead to unsatisfied clause(s) implying a conflict. The search then undoes the current assignment (i.e. backtracks), so that other assignments can be tried. This backtracking process is the basic mechanism for retreating from regions of the search space that do not correspond to satisfying assignments. The search terminates successfully if all clauses become satisfied; otherwise if all possible assignments have been exhausted it terminates without success.

Heuristics: The choice of branching variables largely influences the portion of the decision tree that needs to be explored. Over the years many different branching heuristics have been proposed and evaluated [83, 197]. Heuristics for choosing variables are more or less arbitrary, usually based on some obvious statistics such as clause-length[5] , literal appearance frequency etc. - for example introduced in GRASP [197]. In practice, the solver must search the entire space one way or the other. Therefore, the main research focus on SAT branching heuristics has been to discover conflicts as early as possible. Another principle guiding the design of branching heuristics in SAT is the cost to evaluate a heuristic. Currently, the most successful branching heuristics all have sublinear asymptotic time complexity about the size

[5]The number of literals in any clause in a propositional CNF formula. For an FOL ontology, we formalize this measure as clause-width, and more generally as formula-width for an FOL-CNF formula.

of the formula. Variable State Independent Decaying Sum (VSIDS) implemented in CHAFF [209] is a cheap and efficient branching heuristic. Several other heuristics [115, 235, 78] were later introduced that performed competitively compared with VSIDS. Despite heuristics, sometimes bad decisions can be made in selecting branching variables and this can make the problem much harder to solve. Random restart resets the variable assignment and starts search all over but keeps any previously learned information to guide future search. Fine-tuned restart strategies have led to an increase in robustness of solvers.

Deduction and Pruning: The `DPLL` algorithm iteratively applies the resolution rule among pairs of clauses until either: the empty clause is generated, in which case the original set of clauses is unsatisfiable; or no more resolution inferences are possible, i.e. the problem is saturated, which from theoretical results then means the problem must be satisfiable. At the core of `DPLL` are two satisfiability-preserving resolution-type transformations to simplify the formula so that it contains no trivial clauses[6] .

- ***Unit literal rule*** or unit resolution is applied when the formula contains a unit clause, i.e. a clause with only a single literal. Since the only way to satisfy such clause is to set the adequate value to make that literal true, it is possible to remove all clauses where the literal occurs (which are already satisfied) and remove every occurrence of its complement (which are set to false and do not contribute to satisfy any clause). After applying unit resolution, new unit clauses can be generated allowing the process to iterate and perform even further simplifications. This iterated propagation is known as unit propagation and performed until no unit clauses are left. If an empty clause is generated when performing unit propagation, this is known as a conflict. If a conflict occurs during the preprocessing stage, then the instance is unsatisfiable and we must backtrack. The process of doing assignments in a chain using the unit resolution rule and of detecting conflicts is called Boolean Constraint Propagation (BCP).

[6] Clauses that have a pair of contradicting literals.

- ***Pure literal rule*** is applied when a literal appears in the formula in only one phase (i.e. always positive or always negative). Then it is possible to assign it the truth value that will satisfy all the clauses where it occurs, effectively allowing us to remove all those clauses. After applying this rule, the resulting formula is no longer equivalent, but just equisatisfiable, to the original one. This is particularly important in the context of incremental satisfiability solving, where new clauses added later might invalidate previous applications of this rule. However, this is a costly process compared to any gains provided by the simplification [207, 125] and therefore, most SAT solvers do not use pure literal rules in the deduction process by default.

Equivalence reasoning is another deduction mechanism that uses additional data structures to capture the information that two variables are equivalent to each other (i.e. they must assume the same value to make the formula satisfiable). Li [181] incorporated equivalence reasoning into the satz solver [182] and observed that it is effective on some classes of benchmarks. Additional cases of resolution and simplification, such as subsumption and variable elimination are possible and explained later in Section 3.2.2. Some rules are much costlier to implement, so many researches are concerned with finding a good trade-off between fast algorithms but sophisticated reasoning methods to compute deductions.

These improved DPLL heuristics allow solving SAT problems with very large number of propositional variables but is still laborious when handling the magnitude of variables that result from the translation of FOL ontologies to propositional logic. This motivates the research undertaken in the latter portion of this dissertation, to study ontology measures that influence the quick exponential build-up of variables and identifying a simplification mechanism to slow this growth.

2.3 Automated Reasoning for First-Order Logic

Automated reasoners for FOL, often summarily referred to as Automated Theorem Provers (ATPs) typically support one or more of three fundamental reasoning tasks for problems in

FOL: proving satisfiability, proving entailments (including unsatisfiability), and answering queries. They fall into two categories:

1. **Theorem provers** prove unsatisfiability (inconsistency) of an ontology or, in a similar fashion, prove theorems about an ontology. To prove unsatisfiability they either derive a proof by contradiction or generate an empty clause via resolution. They are widely employed for query answering tasks.

2. **Model finders** prove satisfiability (consistency) of an ontology by generating a finite model if one exists, or report that none exists when it runs into intractability. Model finding is useful to generate models of axiomatizations and countermodels of theorems, which not only helps in consistency verification but often helps in developing interesting mathematical insights [16]. Models are useful for answering questions via model checking, as shown in [46, 40].

Our work in Chapters 5, 6 and 7 are concerned with model finding only, so we focus on discussing model finding techniques and tools here.

2.3.1 Common Algorithms for Finite-Model Finding

There are three commonly adopted approaches to finite-model building for first-order logic: (1) the Mace-style approach [199, 279, 269] works by converting the FOL formula into propositional logic and handing them off to a SAT-solver, (2) the SEM/Falcon-style approach [279, 269, 280, 281] builds a model directly via traditional search techniques, often pruning the search by manipulating the given sentences to take the consequences of the partially built model into account, (3) the Darwin-style approach [23] is similar to the Mace-style approach, in that it reduces a given FOL formula into a problem in the EPR fragment (EPR - effectively propositional logic also called the Bernays-Schönfinkel-Ramsey fragment of FOL [231], where the formula contains no function symbols), a quantifier-free, function-free first-order logic, and then decides satisfiability using a decision procedure. Finite-model finders of the first and last kind build a sequence of translations incrementally[7] over finite domain sizes $1, 2, \ldots$ and then test satisfiability.

Resolution-based as opposed to tableaux-based[8] instantiation methods are commonly used in tools (using the three kinds of model finding paradigms discussed above) that competitively perform in ontology model finding tasks. Model finders that employ these methods convert the ontology for increasing domain sizes to a decidable logic by maintaining a set of instantiated clauses and analyzing it for satisfiability. Paradox [56] and Mace4[9] convert the problems to propositional logic, essentially creating a series of SAT problems of increasing size until a SAT model is found. Others, such as iProver [167] and Darwin-FM [23] use specialized calculi that operate on a conversion to a more expressive function-free clause logic instead of propositional logic. These avert the size and associated memory consumption issues experienced in conversion to propositional logic and are claimed to significantly scale better for higher-arity predicates and for larger domain sizes.

- The **MACE-style** method [199] used in Paradox [56], MACE4 [200], and Vampire [230] transforms the FOL formula into a propositional logic clause set for increasing domain sizes by introducing propositional variables representing the FOL literals. The resulting clause set is then flattened and instantiated for increasing domain sizes, which is then solved by a SAT-solver. Flattening converts a regular FOL clause set into clauses with only shallow literals. A shallow literal does not contain a term that is a not a variable (such a function) and is not of the form $x \neq y$.

- **Inst-Gen** is an instantiation-based method [166] used in iProver [165] that uses instantiation in conjunction with propositional satisfiability checking and redundancy elimination in a modular fashion. Finite-model finding using this method is achieved through translating the problem to the Efficient Propositional (EPR) fragment. The basic idea is the use of a

[7] MACE-style begins search with the lowest domain size $d = 1$, whereas the Darwin style approach begins with an optimal lower bound d based on some analysis of the input clauses.

[8] Their differences are reviewed in [100].

[9] Mace4 propositionalizes the problem but applies a more specialized constraint satisfaction algorithm.

resolution kind inference rule on sets of instantiated premises [109] – a set of FOL clause S is satisfiable iff its propositional abstraction $S \perp$ is satisfiable. Unlike the resolution rule, the Inst-Gen rule does not increase the number of literals in clauses but is also restricted to select literals chosen through a semantic selection function. The number of literals in the generated clauses is further reduced through simplifications such as dismatching constraints, global and propositional subsumption (for both ground and non-ground clauses), blocking non-proper instantiations [166]. The calculus also combines resolution with instantiation to generate additional clauses, which are sometimes useful for simplifications. This is used together with saturation to determine satisfiability or unsatisfiability.

A **saturation** algorithm iteratively applies a set of inference rules to the input set of CNF clauses S to derive new clauses that are added to S. If at some moment the empty clause is obtained, then the input set of clauses is unsatisfiable. If saturation terminates without generating the empty clause, S is satisfiable. If it runs until the system runs out of resources, but without generating the empty clause, then it is unknown whether S is unsatisfiable. Saturation will result in a rapid growth of search space, and this is handled by simplification rules such as clause elimination techniques. Each time a new clause is generated by an inference, the prover decides whether this clause should be kept or discarded. Further inferences are made using only a subset of the kept clauses.

- **Model Evolution (ME) calculus** presented in [26] is a version of instantiation-based methods that interleaves instantiation with propositional `DPLL` style reasoning and was first implemented in the Darwin theorem prover [24]. It uses the FDPLL calculus [21], a variant of DPLL simplifications rules to split, subsume and resolve clauses. It is like MACE-style but differs in the nature of the input formula (function-free clauses vs propositional logic), and the way the size of input clauses grows (linear vs exponential). E-Darwin, which implements the extension of the ME calculus with equality [26], when tested on a TPTP library of FOL formulae placed second after Vampire [25], and FM-Darwin, which converts

FOL formulae to function-free clauses sets was found to be more memory-efficient, but was placed third after Paradox and Mace4 in SAT-based FOL model finding [23].

The ME calculus was found to work best on certain fragments of FOL that proves difficult for other methods, specifically the EPR fragment[10] . Other older or less-used instantiation-based methods such as the Hyperlinking calculus (HL) [179], Ordered Semantic Hyperlinking calculus (OSHL) [221], Confluent Connection Calculus (CCC) [22], disconnection calculus are compared in [180].

2.3.2 State-of-the-art Model Finders Employed in this Dissertation

Far fewer model finders exist than theorem provers [279], as also evident from CADE's automated theorem proving competition (CASC)[11] [262]. The CASC divisions relevant to FOL model finding and their latest winners are:

- FNT - first-order non-theorems: 1^{st} place - Vampire, 2^{nd} place - iProver second (but not so good with equality).
- EPR with EPS subcategory - effectively propositional non-theorems: 1^{st} place - Vampire, 2^{nd} place - iProver (for non-theorems: 1^{st} place - iProver, 2^{nd} place - Vampire). Previously Paradox often won in this category.
- LTB - first-order theorems from large theories, but has no similar model finding category.
- SAT, the category with CNF really-non-propositional non-theorems (with and without equality), is pretty old that was removed after 2009 and was last won by Paradox.

These leading model finders evaluated against benchmarks in CADE-ATP system competitions are very effective for instances generated from formal verification problems where there are almost no datasets involved [222], however we find that reasoning about complex ontologies

[10] See winners in the different categories/fragments at `http://www.tptp.org/CASC/`

[11] Overview: `http://www.tptp.org/CASC/`, Division descriptions: `http://www.tptp.org/CASC/27/Proceedings.pdf`, Last results: `http://www.tptp.org/CASC/27/WWWFiles/ResultsSummary.html`

(such as SFA-FOL that we introduce in Chapter 4) with real-world datasets has been quite challenging using these tools. And provers that have won in the EPR category (which is NEXT-TIME) are not the preferred choice of solvers for NP search problems - such as finite-model computation – which typically uses solvers that are superior in the FNT category. Therefore, we exclusively focus on experimental results from Paradox, iProver and Vampire in Chapter 7, as these ATPs have had fairly consistent success in the verification of FOL ontologies (see, e.g. [127, 169, 168, 239]). These are also the solvers that have won the SAT and EPR categories in the CADE ATP competitions several times [266, 261]. In this section we briefly introduce model finders that are part of state-of-the-art automated reasoners[12] . We use it because it has shown promise in preliminary work [127] and has repeatedly won the SAT division until it was no longer part of the CASC.

Paradox:[13] is a MACE-style finite model finder [56] that employs the MiniSat solver[14] [89] for propositional reasoning. Paradox upgrades the traditional MACE method using four techniques: (1) variable reduction using term definitions, (2) incremental SAT that reuses information such as learned clauses and other heuristic scores for incremental model sizes, (3) static symmetry reduction to eliminate search in isomorphic parts of a search space by adding symmetry breaking formulae, and (4) sort inference for more refined symmetry reduction. This solver uses incremental SAT solving, which was first introduced in the CHAFF SAT solver[15].

iProver:[15] is an instantiated-based solver for classical first-order logic with equality. It is implemented in OCaml and also integrates MiniSat. It is based on a version of the Inst-Gen calculus, DSInst-Gen [166] and uses a combination of superposition and instantiation [82]. iProver encodes the problem in the EPR fragment and passes it to the MinSat solver. The solver is tuned to implement different simplification steps at various

[12]However it is important to note that all these tools are also use for theorem proving tasks. While Paradox is only a model finder, Vampire and iProver function as theorem provers in their default mode, and as a model finder using the casc-sat mode

[13]`https://github.com/c-cube/paradox`

[14]It also allows the integration of any other state-of-the-art SAT solver.

[15]`http://www.cs.man.ac.uk/Ëœkorovink/iprover/`

stages such as forward and backward subsumption, tautology elimination, subsumption resolution, global subsumption for clauses with variables that are semantically guided by literal selection after restarts. It also features state-of-the-art techniques such as indexing, redundancy elimination based on dismatching constraints, blocking non-proper instantiations, and predicate elimination preprocessing. To improve theorem proving performance with large theories, iProver implements an abstraction-refinement mechanism [137] that selects relevant axioms to prove a conjecture based on their syntactic or semantic relationship.

Vampire:[16] [172] uses the superposition calculus (proof search by saturation) for first-order theorem proving, symbol elimination for identifying program properties, and several theory functions on integers, real numbers, arrays and strings (capable of sort and arithmetic) which make it a useful reasoning tool with theories and quantifiers. It also implements a MACE-style finite model builder like Paradox. But while Paradox constructs SAT problems in an incremental fashion and solves them, the SAT solver in Vampire is set to work non-incrementally [230]. This setting helps the use of variable elimination techniques more efficiently. In addition, a superposition-based architecture called AVATAR [273] is incorporated, which helps make '*splitting decisions*' for clauses to reduce the search space.

Unlike the CASC competition, we do not intend to compare model finders against each other, instead through theoretical and experimental evaluation we try to get a better sense of general bottlenecks and scalability of model construction for FOL ontologies with data.

2.4 Ontological Formalization of Space

Ontologies of space formalize spatial concepts and relations that describe an object's location with respect to its surrounding space and to other objects. This includes: (1) topological (e.g. connected) and mereological relations (e.g. inside), (3) absolute location (e.g. geometry with coordinates) (4) orientation (e.g. south, southwest), (5) distance from other objects, (6) fuzzy relations (e.g. close, far) and so on. These relations may capture

[16] `https://vprover.github.io/`

qualitative (which includes mereological or/and topological) information or quantitative (metric) information. Human spatial expressions often rely on qualitative more than quantitative spatial information. Ontologies with spatial relations are traditionally modeled from a linguistic perspective [174] or a formal perspective or a combination of both[17] . Linguistically motivated spatial relations focus on prepositions and are modeled from a reference frame relative to the user. They do not provide spatially explicit, computational semantics for the relations and are open to multiple possible spatial interpretations, for example the relations *in* and *on* from [68], and relations in the GUM-Space ontology. Formal spatial relations are based on some mathematical formalism such as a calculus, - e.g. Double cross calculus to represent orientation relative to axis [243], Du's logic of near and far (LNF) to represent proximity [81], and RCC to represent connectivity between regions [63]. Mereotopological relations are among the most common qualitative spatial relations, and include purely topological relations such as contact/connection or disconnection, and purely mereological relations such as parthood, containment, or inside, as well as relations that describe the interaction of topology and mereology such as overlap (i.e. contact via sharing a part). Many of these relations have also been incorporated into virtually all upper ontologies such as BFO [120], DOLCE [198], GFO [20], Cyc [72], although they may not fully axiomatize the detailed semantics. Besides questions about an object's mereological and topological relations, other concerns that are addressed by these ontological formalizations are questions concerning the relationships between spatial geometries and physical entities, composition/material of the entity. Some detail on the ontological arguments about these formalizations is available in [17]. Refer [132] for mereotopological theories and relations.

2.4.1 Qualitative Spatial Representations

Within spatial information science, there are several approaches to the formalization of qualitative spatial representation (QSR). Qualitative calculi (based on some constraint

[17] Note that topological relations can be linguistic or formal.

satisfaction criteria) such as the Region Connection Calculus, RCC proposed by Randell, Cui & Cohn [226] and others in [44, 62] categorize space as a set of n-dimensional regions; topological constraints are based on point-set intersections such as 9-intersection relations [57, 59, 91, 92, 203]. The 9-intersection method [91, 92], its dimension-extended refinement (DE-9I) [57] and extensions thereof [60, 202, 238] determine mereotopological relations between geometric data by computing a matrix of values that indicate the pairwise intersections of two object's *interior* ($\circ$), *boundary* (∂), and *complement* ($'$). Each of the nine pairs have either Boolean values – empty nor non-empty intersection – as in the original 9-intersection framework [91], or have dimensional values – either -1 (empty intersection), 0, 1, or 2 – as in the dimension-extended method.

Then there are axiomatic treatments of mereotopology (refer to Section 5 in [132]), which constrain the interpretations of one or two primitive relations, such as contact and/or parthood, and define other relations, such as overlap or external contact, in terms of the primitive ones [52]. These ontologies formalize relations between geometric entities that have the same dimension [20, 52, 65, 223, 227, 251], and some others between multidimensional spatial entities that can coexist. Geometry in multidimensional theories is defined entirely in terms of mereotopological relations, including work by Galton [107], Gott's INCH Calculus [118], and the CODI ontologies [128]. CODIB builds on and extends the theory CODI (which doesn't include any notion of boundaries) [130, 128] by the additional relation of boundary containment. Unlike other multidimensional theories [107, 251], CODI and CODIB allows entities of lower dimensions to exist independent of entities of higher dimension, similar to how such entities (e.g. polylines or points) are used in geometric data standards. [107, 251] require each line or curve to be part of the boundary of some 2D region and each point to be the endpoint of some curve in a model. The INCH calculus [118], on the other hand, does not model boundaries at all. Another alternative formalization of multidimensional mereotopology is provided by the GFO space ontology [20] that is part of the General Formal Ontology (GFO). However, GFO space is primarily concerned with *physical*, *phenomenal space* (i.e. the space of material objects), which is different from the kind of *abstract*, *extensional space* that geometric data models describe[18] [127, 20].

2.4.2 FOL Ontologies for QSR: CODI, RCC, INCH

Axiomatic ontologies of mereotopological relations combine mereological relations (i.e. parthood) and topology (i.e. connectedness), which allows defining finer spatial relations such as incidence (i.e. 1D and 2D region connected via a shared part). The utilized primitive relations include *Parthood*, *Connection*, *Simple-Region*, *Congruence* in [44]; *Connection*, *Part*, *Convex hull* in [225]; *Part*, *Boundary*, *Located-at* in [253]; *Containment*, *EgDim*, *LessDim*, *ZEX* in [128]. RCC is the most popular unidimensional theory [227], while CODI and INCH are multidimensional theories, which motivated us to choose these formalizations for our model finding experiments. Moreover, CODI is already verified and used, which is why we extend CODI with Simple Features in Chapter 4. In this section we present an overview of the COntainment-Dimension (CODI) ontology [130, 128], the RCC-FOL ontology [1], which is a bare formalization of the RCC-8, and finally we discuss the INCH calculus [118]. We introduce only those FOL-predicates (concepts and relations) that we use for the formalization of the Simple Feature standard in Chapter 4 and those included for the theoretical and empirical analysis of SAT-based FOL model finding presented in Chapters 5 and 6. The variety of existing axiomatic theories are more thoroughly reviewed in [132].

2.4.2.1 COntainment DImension Ontology

Here we review CODI axioms that are generically used in model finding experiments in Chapter 6 and used in formalizing Simple Features in Chapter 4. We discuss additional details of CODIB (the boundary-extended version of CODI) in Chapter 4, where it is more relevant. CODI axiomatizes mereotopological relations in a dimension-independent way using two primitive relations: (1) the mereological notion of containment, $Cont(x, y)$, and a

[18] For example, in phenomenal space, any road would be a 3D object, whereas in abstract space it is typically modeled as a 1D spatial feature.

relation $\leq_{\text{dim}}(x, y)$, read as "x has the same or a lower dimension than y", to compare the dimension of two entities [128, 130]. In addition, the primitive unary predicate $S(x)$ is used to denote spatial regions, which captures mathematical regions of space whose existence is independent of whether an actual physical object occupies a spatial region or not. *Cont* is reflexive, symmetric, and transitive (Cont-A1–A3) and allows defining the zero (i.e. null) region denoted by the unary predicate *ZEX* (ZEX-D). Containment requires the contained entity to be of the same or a lower dimension than the entity it is contained in (CD-A1).

The relative dimension $\leq_{\text{dim}}(x, y)$ alone can define additional relations of equal dimension $=_{\text{dim}}(x, y)$, lesser dimension $<_{\text{dim}}(x, y)$, minimal dimension $MinDim(x)$ (i.e. the dimension of a point; D-D6), and next-lower dimension $\prec_{\text{dim}}(x, y)$ (D-D7). The relation $\leq_{\text{dim}}(x, y)$ is axiomatized to form a discrete (i.e. there is a next-lower dimension for every non-minimal entity) and bounded (i.e. a lowest and highest dimension exists) pre-order over all spatial regions (axioms Dif-A2, Dif-A3a–c, Dif-A4 in [128], but are omitted here because they are not used in our study). This also implies that every spatial region must be of uniform dimension, i.e. all components (i.e. parts) thereof are of the same dimension, precluding objects such as a region consisting of a 2D region and a separate, isolated point or linear feature. Spatial regions can still *contain* lower-dimensional entities (e.g. a 2D region containing 1D features and points). Using the relative dimension of the involved entities, containment is specialized to parthood (i.e. equidimensional containment; EP-D) and proper parthood (EPP-D). Minimal spatial entities have no proper parts (ME-D2), that is, they are indivisible. There can be minimal entities within each dimension. See [128] for the full details of the axiomatization.

(Cont-A1) $S(x) \wedge \neg ZEX(x) \leftrightarrow Cont(x, x)$

(***containment is reflexive for all nonzero spatial regions***)

(Cont-A2) $Cont(x, y) \wedge Cont(y, x) \rightarrow x = y$ (***containment is antisymmetric***)

(Cont-A3) $Cont(x, y) \wedge Cont(y, z) \rightarrow Cont(x, z)$ (***containment is transitive***)

(ZEX-D) $ZEX(x) \leftrightarrow S(x) \wedge \forall y[\neg Cont(x, y) \wedge \neg Cont(y, x)]$ (***zero region***)

(CD-A1) $Cont(x,y) \rightarrow x \leq_{\text{dim}} y$ ***(interaction between** $Cont$ **and** $\leq_{\text{dim}}$**)***

(D-D6) $MinDim(x) \leftrightarrow \neg ZEX(x) \wedge \forall y\, [\neg ZEX(y) \rightarrow x \leq_{\text{dim}} y]$ ***(minimal-dimensional entities)***

(D-D7) $x \prec_{\text{dim}} y \leftrightarrow (\leq_{\text{dim}} y \wedge \neg(y \leq_{\text{dim}} x) \wedge \forall z\, [z \leq_{\text{dim}} x \vee y \leq_{\text{dim}} z]$ ***(next-lower dimension)***

(EP-D) $P(x,y) \leftrightarrow Cont(x,y) \wedge x =_{dim} y$ ***(parthood: equidimensional containment)***

(EPP-D) $PP(x,y) \leftrightarrow P(x,y) \wedge x \neq y$ ***(proper parthood)***

(ME-D2) $Min(x) \leftrightarrow \neg ZEX(x) \wedge \forall y\, [\neg PP(y,x)]$ ***(minimal entities within a dimension)***

Contact, $C(x,y)$, as the most general topological relation is definable as x and y sharing some contained object (C-D) and is provably reflexive and symmetric. Specialized types of contact can be distinguished based on the relative dimension: partial overlap $PO(x,y)$ holds only between entities of equal dimension and requires them to share a part (PO-D); incidence $Inc(x,y)$ holds between entities of different dimension and requires a part of the lower-dimensional entity to be shared with the higher-dimensional entity (Inc-D); and superficial contact $SC(x,y)$ requires the shared entity to be of a lower dimension than both of the entities in contact (SC-D).

(C-D) $C(x,y) \leftrightarrow \exists z[Cont(z,x) \wedge Cont(z,y)]$ ***(contact)***

(PO-D) $PO(x,y) \leftrightarrow \exists z[P(z,x) \wedge P(z,y)]$ ***(overlap in a part)***

(Inc-D) $Inc(x,y) \leftrightarrow \exists z[(Cont(z,x) \wedge P(z,y) \wedge z <_{dim} x) \vee (P(z,x) \wedge Cont(z,y) \wedge z \prec_{\text{dim}} y)]$

(incidence)

(SC-D) $SC(x,y) \leftrightarrow \exists z[Cont(z,x) \wedge Cont(z,y)] \wedge \forall z[Cont(z,x) \wedge Cont(z,y) \rightarrow z \prec_{\text{dim}} x \wedge z \prec_{\text{dim}} y]$ ***(superficial contact)***

While CODI does not distinguish different primitive types of entities, they can be defined: *PointRegion*s (which encompass individual points and sets of points) are of minimal dimension, *Curve*s are of next higher dimension, and so forth [129]. All of these primitive classes specialize the class S of abstract spatial regions.

(PR-D) $PointRegion(x) \leftrightarrow S(x) \wedge MinDim(x) \wedge \neg ZEX(x)$ ***(point sets)***

(Point-D) $Point(x) \leftrightarrow PointRegion(x) \wedge Min(x)$ ***(individual points)***

(Curve-D) $Curve(x) \leftrightarrow S(x) \wedge \forall y[PointRegion(y) \rightarrow y \prec_{\mathrm{dim}} x]$ ***(curves as 1D entities)***

(AR-D) $ArealRegion(x) \leftrightarrow S(x) \wedge \forall y[Curve(y) \rightarrow y \prec_{\mathrm{dim}} x]$ ***(areal regions as 2D entities)***

Clarification: Axioms about the mereological operators (intersection, difference, complement and sum of entities) from [128] are not included in our experiments in Chapters 5 and 6.

2.4.2.2 The RCC Ontology:

The axiomatization of the Region-Connection Calculus (RCC) theory by Randell, Cui and Cohn [227] uses the primitive connectedness relation, $C(x, y)$, which is a reflexive and symmetric relation (RCC:A1,A2) as the basic element to define a set of mereotopological relations between pairs of equi-dimensional regions. RCC-8 contains eight jointly exhaustive pairwise disjunct (JEPD) binary relations, but the axioms in the ontology used in our work only formalizes five of these relations (P, PP, O, EC, $NTPP$): $P(x, y)$ - 'x is a part of y' (RCC:D1); $PP(x, y)$ - 'x is a proper part of y' (RCC:D2); $O(x, y)$ - 'x overlaps y' (RCC:D3); $EC(x, y)$ - 'x is externally connected with y' (RCC:D4); $NTPP(x, y)$ - 'x is a non-tangential proper part of y' (RCC:D5). The axioms stating the relational operations sum, product, universal element and complement are not included in model finding experiments in this dissertation. These axioms are available in the COLORE repository[19] . Although the RCC's DC relation is not formalized, we can easily represent this notion as negated connectedness ($\neg C$), as we will use this to represent disconnected objects when we write data assertions for datasets used in Chapters 5 and 6.

(RCC:A1) $C(x, x)$ ***(connected is reflexive)***

(RCC:A2) $C(x, y) \rightarrow C(y, x)$ ***(connected is symmetric)***

[19]https://github.com/gruninger/colore/tree/master/ontologies/mereotopology/

(RCC:D1) $P(x,y) \leftrightarrow \forall z[C(z,x) \rightarrow C(z,y)]$ ***(parthood)***

(RCC:D2) $PP(x,y) \leftrightarrow P(x,y) \wedge \neg P(y,x)$ ***(proper parthood)***

(RCC:D3) $O(x,y) \leftrightarrow \exists z[P(z,x) \wedge P(z,y)]$ ***(overlap)***

(RCC:D4) $EC(x,y) \leftrightarrow C(x,y) \wedge \neg O(x,y)$ ***(external connection)***

(RCC:D5) $NTPP(x,y) \leftrightarrow PP(x,y) \wedge \neg existsz[EC(z,y) \wedge EC(z,y)]$

(non-tangential proper parthood)

2.4.2.3 The INCH Ontology

The INCH ontology [130] is based on the INCH calculus initially formalized in [118], and has five primitive relations: a dimension-independent mereological primitive: $INCH(x,y)$, with the intended meaning '*x includes a chunk of y*' is a more expressive version of RCC's C (I:PA7); $CH(x,y)$, where a chunk denotes an equi-dimensional part (I:D4); $CS(x,y)$ denotes x as a constituent of y if they $INCH$ a common spatial extent (I:D4); $ZEXI(x)$ denotes the region x with zero extent (I:D6); $GED(x,y)$ denotes that the dimensionality of x is at least that of y. The dimensional primitive $GED(x,y)$ is defined such that y is a zero-region or using the containment relation $INCH(x,y)$ to indicate x is greater or of equal dimension to y (I:D7). In addition INCH contains two other predicates defined using the primitives: $OV(x,y)$ denotes that the two extents x and y $INCH$ each other (I:D2); $CO(x,y)$ denotes that the two extents x and y are connected (I:D3) – this is similar to partial overlap and incidence in CODI. [130] provides a formalization of the INCH calculus using mereotopological primitives from CODI ($Cont$ and P).

The ontology includes axioms formalizing the properties of transitivity, reflexivity and extensional properties for $INCH$ and GED (I:PA1-PA6). We refer the reader to the COLORE repository[20] for these additional axioms.

(I:D1) $CS(x,y) \leftrightarrow \forall z[INCH(x,z) \rightarrow INCH(y,z)]$ ***(constituent)***

[20]https://github.com/gruninger/colore/tree/master/ontologies/inch

(I:D2) $OV(x,y) \leftrightarrow \forall INCH(x,y) \wedge INCH(y,x)$ ***(overlap)***

(I:D3) $CO(x,y) \leftrightarrow \forall z[\neg ZEXI(z) \wedge CS(z,x) \wedge CS(z,y)]$ ***(contact)***

(I:D4) $CH(x,y) \leftrightarrow \forall INCH(x,y) \wedge \forall z[(INCH(x,z) \wedge INCH(z,x)) \rightarrow (INCH(y,z) \wedge INCH(z,y))]$

(chunk - equidimensional part)

(I:D6) $ZEXI(x) \leftrightarrow \neg INCH(x,x)$ ***(zero region - no entity is contained in ZEXI)***

(I:D7) $GED(x,y) \leftrightarrow ZEXI(y) \vee \exists z[INCH(x,z) \wedge INCH(z,y)]$ ***(greater or equal dimension)***

(I:PA7) $INCH(x,y) \leftrightarrow \exists z[CS(z,x) \wedge CH(z,y)]$ ***(requires a chunk of x to overlap with y)***

2.4.2.4 Summary of Formalizations used in our Model Finding Studies

Ontology	Signature	Number of relations		Total axioms (including definitions)
		Unary	Binary	
CODI	Cont, Leq, S, ZEX, Lt, Gt, Geq, EqDim, Covers, P, MinDim, MaxDim, PointRegion, Point, Curve, ArealRegion, PP, PO, Inc, SC	8	13	31
RCC	C, P, PP, O, EC, NTPP	-	6	8
INCH	INCH, GED, ZEXI, CH, CS, CO, OV	1	6	16

Table 2.1: Summary that explicitly lists the signature, and contains a statistic of the number of relations (unary, binary), and axioms included, for CODI, RCC, and INCH that are used in our model finding experiments.

CHAPTER 3

RELATED WORK

The two overarching contributions of this dissertation are: firstly, enabling integrated and stand-alone geometric-qualitative spatial reasoning, secondly, improving the scalability of model finding using FOL ontologies with moderately-sized datasets. Through extensive literature review we identified the gaps and inadequacies existing in current state-of-the-art tools and methods, which also inspired us to embark on this work. In this chapter we present some of this relevant work, which sets the context for, and motivates the rest of this dissertation.

In Section 3.1 we give an account of work conducted in reasoning with FOL ontologies - for verification and other purposes, with a focus on reasoning tasks commonly undertaken, scalability achieved, and limitations of current tools. We also briefly review how our work differs from related research that has studied the tractability of SAT solving mostly with original propositional logic problems in Section 3.2, but does not touch on the hardness of SAT solving on FOL problems. We present a survey of some of existing formula simplification techniques for general SAT in Section 3.2.2 and those specific to FOL in Section 3.2.3. Finally in Section 3.3, we discuss work that has been done for performing qualitative and quantitative spatial reasoning using formal ontologies. It must be noted that the list of related work given in this chapter is not an exhaustive one as the field is a rapidly evolving one, and new tools with advanced algorithms and heuristics, are developed on an ongoing basis.

3.1 Reasoning with FOL Ontologies

Extensive development of formal ontologies has generated considerable research in advancing automated reasoning techniques and tools called ATPs (theorem provers and model finders – cf. Section 2.3) that help with reasoning tasks such as query answering, proving theorems and ontology consistency checking. Theorem provers determine the unsatisfiability of an ontology either by deriving a proof by contradiction or generating an empty clause via resolution. In a similar fashion, they can be used to prove theorems about an ontology, and for query answering tasks. Model finders prove satisfiability of an ontology by generating a finite model if one exists [16, 46, 40]. However, the expressiveness of the FOL language combined with the complexity of SAT reasoning often impedes efficient reasoning, model finding having been found in practice to scale even less than theorem proving, most often quickly becoming intractable once moving beyond very small domain sizes. In this section we discuss some of the existing work on studying the practical limits of reasoning with FOL ontologies.

3.1.1 Theorem Proving with FOL Ontologies

Much work in first-order reasoning has focused on theorem proving. Even within theorem proving, most works use small axiomatizations that contain very few functions or predicates – such as theories in mathematics [248, 42], axiomatizations for software and hardware verification [55, 217, 171], and software design [242]. Similar lines of work include evaluation of Vampire extended with Boolean sorts (Vampire with FOOL) on theorem proving based verification problems [169], verifying properties of cloud networks using Vampire as a theorem prover [168], theorem proving for data model verification in FOL using Spass and Z3 [41]. All these works use little to no data/facts. There are also the kind of benchmark problems that are included in the TPTP library[1] [260, 264, 263] that theorem provers are evaluated against in the annual automated theorem proving competition (CASC)[2] . Query answering with larger vocabularies again mostly employ theorem provers rather than model finders, for example: comparison of Darwin, Vampire, Epilog in [145], theorem proving using Vampire with SUMO, a large ontology containing about 1000 terms and 4000 sentences [218], theorem proving using Vampire, SPASS and E for query answering on the first-order version of Cyc KB containing 1,253,117 sentences [224], using OTTER theorem prover in developing expert medical reasoning systems, though on relatively small problems [190].

[1] `tptp.org`
[2] `http://www.tptp.org/CASC/`

Most ATPs combine theorem proving and model finding capabilities. Evaluation of these competitive tools is essentially done through comparison analysis of their theorem proving performance. For example, the strength of Vampire, a consistent top winner in the CASC ATP competitions since 2000 is evaluated primarily through proving theorems as discussed in [270, 212]. In rare cases, when the model finding performance of ATPs are evaluated, datasets are not used, for instance experimental assessment of Paradox, the consistent winner of the SAT category until 2000 compared in [232].

3.1.2 Scalability of Model Finding for FOL Ontologies

General first-order satisfiability is undecidable, but with efficient heuristics modern ATPs can build finite models for small-sized problems [239, 46]. The major hurdle for improving this scalability with larger decidable problems is intractability because available algorithms to solve them have exponential time complexity [110, 206]. Existing SAT solvers are typically good at determining unsatisfiability [241], but when a problem is theoretically satisfiable, many solvers cannot find a solution, i.e., a model, either because the algorithm fails to find a solution, or due to hardware limitations, where the system runs out of time or memory, which typically results from an extremely large search space. Theorem provers scale rather badly with large problems, but the challenge for model finders is even higher. ATPs that perform SAT-based FOL reasoning incorporate state-of-the-art standalone SAT solvers. While many works claim the impressive performance of SAT solvers on industrial problems containing millions of variables [39, 272], this success has been facilitated by the fact that SAT has very simple syntax and semantics. Unfortunately, SAT provides a poor modelling language, and many domains such as geosciences, require a more expressive formalization in first-order logic using predicates and functions and not just propositional variables. Reduction of an FOL problem to a SAT problem drastically increases the complexity of the problem through the addition of additional variables and predicates during the process of clausification, flattening and skolemization, and an exponential increase in search space based on the number of individuals in the domain.

So far, in practice, off-the-shelf model finders haven't been able to generate models with domain sizes larger than about 20 [46] – tested with Paradox and MACE 2.0, which effectively is a limitation in domains such as GIS where even with a very small dataset, the domain size in the ABox is very high. In the absence of ground facts (i.e., the ABox), model finding can be efficient for very large ontologies [218], but is only aimed at finding the smallest model and does not serve the purpose of tasks such as data-driven ontology verification or identifying datasets that satisfy an axiomatization set. The performance of model finders Paradox (generated models of upto size 5) and Darwin (timed out for most cases) was found to be considerably lower compared to the promising results exhibited by Vampire and iProver for theorem proving experiments conducted on the FOL translation of OWL2's Full semantics (consisting of 558 axioms) with a test datasets [239]. Other ATPs such as FM-Darwin that use more efficient theory translations such as function-free clause logic[3] also does not scale to generate models larger than 20. The comparison study in [23] showed FM-Darwin capable of constructing models upto size 10, while Mace4 failed at size 7, and Paradox became intractable from size 7 onwards for the same problems in SAT with more than $8 \cdot 10^5$ variables and $5 \cdot 10^4$ clauses. Moreover, model finders are almost always evaluated against the TPTP problem library [260, 264, 263], the standard benchmark problems used in the CADE-ATP competitions, which do not reflect the scale and complexity of reasoning encountered in data-driven model finding using spatial ontologies such as CODI, RCC and INCH.

3.2 SAT-Based Model Finding for FOL Ontologies

Propositional SAT solvers are employed in many FOL theorem provers (Otter and Prover9) and model finders (Paradox, Vampire), mainly reasoners that adopt the MACE-style approach. The significance of the SAT problem in studying complexity and for industrial reasoning tasks

[3]Problem size grows much slower compared to the exponential growth in propositional logic and therefore does not require as much memory as solvers that translate to SAT.

has spurred many SAT algorithm optimizations and hardware acceleration to handle the large amount of computation involved. Current SAT solvers exhibit impressive performance on many industrial problems containing millions of variables [39, 272]. The performance improvement of these solvers based on hardware is still limited based on two factors[250]: first complex algorithms built to handle large and complex SAT problems in the real-world require large RAM and advanced processors, secondly the scale of model finding problems increases exponentially with domain size. Large-scale industrial SAT problems generally have millions of variables, and ten millions of clauses. Therefore, the storage of these variables and clauses has become a resource-intensive bottleneck for SAT solvers that use complete decision procedures[4] .

3.2.1 Studies on Tractability of Propositional SAT Solving

Given a CNF formula F, it is called a k-SAT formula if each clause in F contains exactly k literals and contains unique variables and literals within each clause. Several researchers have investigated the relationship between variables, clauses and algorithmic properties of the random k-SAT search space [53, 215, 85, 5]. Results have led to efficient heuristics such as efficient variable assignment [114] and clause simplification strategies (discussed in Sections 3.2.2 and 3.2.3). SAT is considered exponential in the number of variables [73, 158], i.e., $O(2^n)$ time, where n is the number of variables in the given formula. While 1-SAT and 2-SAT are both solvable in polynomial time, from 3-SAT onwards the complexity becomes exponential [116]. The worst-case 3-SAT algorithm runs in $\mathcal{O}(2^n * t)$ time (improved to $\mathcal{O}(1.5045^n * t)$ in [175]), since each of the 2^n possible truth assignments to n variables requires at most t time to check.

SAT Phase Transition: Numerous studies [53, 206, 205, 4, 84, 3, 104] show the relationship between the empirical hardness and satisfiability of random k-SAT problems to its clause density r (clauses-to-variables ratio), based on experiments conducted on

[4] If the problem is unsatisfiable, then given enough time and space the solver will eventually find a refutation - widely used in all CDCL solvers.

problems following uniform random distributions. This is explained as the 'satisfiability phase transition' phenomenon [195, 71, 61, 53, 117, 136] that divides the solution space of satisfiability problems into three regions that follow an easy-hard-less-hard runtime pattern. This pattern is characterized by the constrainedness of the problem, represented by the clause density. The low-density region constitutes the under-constrained problems with a small number of constraints (or clause set), which in the random case appear to be easy. Because they generally have many solutions, search algorithms have a higher probability of finding a solution and typically have a polynomial running time. The over-constrained problems (typically with a density above 4.6) are problems with a very large number of constraints that also appear to be easy, because intelligent algorithms will generally be able to quickly find a contradiction in the form of an empty, i.e., unsatisfiable clause. Traditional `DPLL` tends to have fast (some times polynomial) performance on SAT instances of these regions. The critically-constrained problems are the hard problems typically with a density ranging from 3.8 upto 4.6. They have few solutions but lots of partial solutions and has exponential runtime performance. This point is referred to as the crossover point [246, 207], where solver performance is the worst. [205] provides a calculation of the satisfiability threshold ratios[5] r_k for different k values (cf. Table 3.1) obtained from random k-SAT problems. It is also to be noted that most of this empirical research has been performed with randomly generated SAT problems, which focus on uniform random distributions, where each variable takes part in a clause with the same probability, and clauses are uncorrelated. However SAT instances that result from the compilation of real-world problems hardly satisfy the pattern of uniform random distributions. Instead, [159] studied easy-to-hard transition in problem hardness as their constrainedness is varied when clauses are dependent as they are typically with real world instances. However, phase transition phenomenon is also solver dependent – for example [50, 105, 208] show linear median running time for problems in the low-density region, and [70] shows the SAT solver Tableau having an exponential runtime for the density

[5]The ratio around which satisfiability problems transitions from easy to hard to easy.

k	1	2	3	4	5
r_k	4.267	9.931	21.117	43.37	87.79

Table 3.1: Satisfiability threshold values for random k-SAT.

4.26. [210] found Reduced Ordered Binary Decision Diagrams (ROBDDs) to perform very well on over-constrained problems. Survey propagation algorithms[6] performed considerably better than `DPLL` when clause density ranged from 4.7 - 5.5 [255], based on experiments conducted on randomly generated problems. In practical FOL ontology reasoning tasks, it is hard to constrain the clause-variable ratio (as we find in our experiments in Chapters 6 and 7, rather we can constrain the signature of the ontology or limit the number of distinct individuals and assertions between them), nor is it very predictive of the performance, rather hardness depends on the absolute values of the number of variables and clauses.

These complexity studies, while they provide interesting insights of specific measures of problem hardness, do not supply any mitigating measures to improve tractability for practical model finding tasks that originate from real ontologies and associated data. Such as, to verify the consistency of a dataset of domain size d against a theory, it is impossible to appropriately control the number of clauses and variables in order to make it easy for a solver. Furthermore, problems with exceedingly large absolute number of clauses may be theoretically easy with appropriate r values but may still be practically infeasible because there is much more redundancy in these clauses as compared to random SAT problems..

Other SAT Heuristic Measures: In addition, in propositional SAT, there are works that exploit domain-specific knowledge to help prune search spaces. For example, signal correlation between nodes of a Boolean circuit has been used to derive good branching and learning heuristics for verification of logic circuits [189]. In SAT, the theory of *parametrized complexity* [98, 67] attempts measuring the complexity not only by the size of the input, but also in terms of a numerical parameter that depends on the input in some way – for example structural graph parameters such as tree-width, branch-width, and clique-width [267]. Other

[6] Heuristic SAT techniques that incorporate a message passing algorithm [48].

structural properties of the formula, ususally captured through a graph[7] , and length of the formula (as measured by the number of literals in the formula) are well-studied measures for characterizing problem hardness and for developing efficient heuristics for decomposing and pruning the SAT search tree [126, 31, 87]. The number of acceptable or satisfiable solutions for a SAT-problem [228] is another relevant factor that determines the hardness of the problem, since SAT instances with few solutions are likely to be harder to solve (i.e., to find any solution at all) than those with a large number of solutions. Practically, the number of solutions to an instance of a SAT problem can vary greatly, and again this correlates closely with the clause-variable ratio [3]. For small ratios, there are many solutions and for large values - from where the typical phase transition occurs - there are few, or even none. To overcome the space-time overhead caused by depth-first traversal and backtracking in the search tree, improvements made to classic `DPLL`-solvers include search space pruning techniques such as early termination [36], unit clause heuristic [182], pure literal heuristic [37], and use of efficient data structures [191, 8]. The process of backtracking search adds clauses to the formula in order to block searching in subspaces that are known to contain no solution. These additional clauses, called blocking clauses (also called learnt or conflict clauses), block solutions that were already found. However, while the addition of blocking clauses prevents repetitions in solution creation, it also significantly inflates the size of the overall number of clauses that need to be tracked and propagated at each step of the search. Thus, the solver slows down in accord with the number of blocking clauses that are added. Eventually, if too many clauses are kept, the solver may exhaust the available memory and terminate. This led to the development of memory efficient algorithms (such as the reachability algorithm implemented in Chaff [122]) without adding blocking clauses, thereby minimizing the space requirements of a solved instance.

Empirical evaluation of different SAT algorithms on a comprehensive suite of benchmarks (with variable count ranging from 10 to 10^6, and clause count ranging from 10^2 to 10^7)

[7]For example, constraint and variable redundancy [216], modularization of the axiomatization [201],symmetry [7] to name a few.

from a range of different application domains showed that conflict-driven clause learning (CDCL) solvers can generally handle problem instances with several million variables and clauses [160, 27, 196, 249]. Many off-the-shelf FOL ATPs employ solvers that follow the CDCL architecture – the default SAT solver in Vampire, Paradox and iProver is MiniSat, which is CDCL based. However, modern solvers still fail, unpredictably, on many practical problem instances. Sources of intractability arising from translating an FOL ontology to a propositional SAT problem is something that is typically not studied in work analyzing the hardness of SAT solving. In particular, SAT heuristics are usually not concerned with the arity of 'FOL-literals', which in practice lead to an exponential increase in the number of propositional variables for the resulting SAT problems.

Graph representations for SAT problems: SAT solvers use graph based representations[8] of the CNF formula to solve a problem instance. The efficiency of determining satisfiability depends on the decompositional parameters for the graph such as treecut-width, tree-depth and pathwidth [108, 186], clique-width [144], branchwidth [185]. Most SAT algorithms have a running time exponential in the tree-width of the graph of the CNF formula, that runs in exponential space or best case polynomial space [13, 6, 77, 103, 237]. For SAT instances the tree-width of a CNF formula is the smallest width for which the clauses in the formula can be arranged in the form of leaves of a rooted binary tree [234]. A rudimentary graph structure has vertices as variables and the edges are representative of clauses – thus the connectivity is determined by the set of clauses and their width. The notion of tree-width is defined via tree decompositions (refer [38, 126] for more information on tree decomposition heuristics). The depth of every decomposition is the largest number of nodes on a path between a root and a leaf. The tree-depth is the minimum depth over all possible tree decompositions of the SAT problem. In saturation-based proof search we're trying to assign truth values to each variable, thus eliminating options from clauses until we either get an empty clause – which indicates a conflict since this particular clause is not satisfiable – or a solution - a

[8]Different graph representations of SAT instances have been proposed in the literature, e.g., incidence graph, primal graph [268], resolution graphs [106] or implication graphs.

complete assignment of truth values to all variables in the propositional CNF formula. The combination of clause selection and variable selection steers this process. The standard backtracking algorithm explores partial variable assignments in a depth first manner to search for a satisfying assignment. This search process is influenced by the heuristics[9] employed for choosing a clause and a variable therein and how efficiently the search tree is pruned or short learnt clauses are remembered. We are unable to find any existing studies that compare the size of an axiomatization (and its sentences) to the tree-width of its SAT-graph structure, while in our work we try to study the study the impact of the complexity of the axiomatization on the increase in size of its CNF translation and the resulting SAT problem, and identify a way to bypass this to some degree.

3.2.2 Simplification Techniques for Propositional SAT Solving

SAT solvers have reached a high level of maturity during the two last decades primarily through efficient heuristics and CNF simplification strategies. The size of CNF formulas in the context of formal verification for typical industrial and real-world SAT problems is often very large, and in practice, the runtime of a SAT solver is very much related to the size of this formula. Clause learning implemented in modern CDCL solvers has clear advantages, but also affects scalability when the learnt clause set is large, taking up memory. To push the limits of tractability, modern solvers often use dedicated *simplification* methods to reduce the search space, and also minimize the number of backtracks. Simplification techniques include *preprocessing* methods [146, 161, 14, 12, 259, 112, 157, 88] performed before search or *inprocessing* methods [156] performed during search, for a substantial decrease in size of the CNF formula. Simplification techniques for CNF formulas is well explored [139] and has been successfully integrated into several SAT solvers such as MiniSat [90], Chaff [209], Glucose [11], and Lingeling [35]. Some well implemented preprocessing techniques include addition or elimination of redundant clauses, clause subsumption and its variants, variants of bounded

[9]Modern SAT solvers use heuristics to select decision variables, the variables that result in highest number of unit propagations.

variable elimination, formula partitioning [194]. They aim mostly at pruning the number of clauses, literals and variables in the input formula.

Clause elimination procedures [14, 33, 47, 88, 192] are special class of resolution-based CNF simplification techniques to remove redundant clauses from CNF formulas resulting in satisfiability-preserved formulas. Implemented at different levels in all winners of the SAT competitions[10] , these procedures have proven to effectively improve solver efficiency [141, 54, 259]. A redundant clauses is either a tautology[11] , a blocked clause[12] or a subsumed clause[13] . Refer [156] for notions and more on types of redundant clauses. Some of the most popular clause reduction procedures include:

- *Subsumption* involves eliminating a larger clause from a formula when it subsumes another smaller clause. A clause A is said to subsume another clause B, iff all the literals in A also occur in B (i.e., $A \subseteq B$). The subsumed clause, B in this case, is redundant and can be removed from the formula. Detection of subsumed clauses is costly, but there are some efficient techniques such as the signature-based algorithm [244].

- *Self-subsumption* is resolution (see Def. 9) with subsumption, applied when a clause A almost subsumes another B, in the sense that all the literals in A are in B except for one. Then their resolvent, $R(A, B)$ is a subset of A, and we can replace A with $R(A, B)$.

- *Bounded resolution* adds all resolvents of size bounded by some function of the formula parameters.

- *Bounded variable elimination* [88, 259] first chooses a variable to eliminate and then removes all clauses containing this variable from the formula while adding to the formula all resolvents of those clauses with respect to the variable chosen. To prevent exponential

[10] `http://www.satcompetition.org/`
[11] It contains two complimentary literals L and $\neg L$.
[12] A clause C is blocked in a formula F if all resolvents upon one of its literals are tautologies.
[13] There exists another clause D and a substitution λ such that $D\lambda \subseteq C$.

blow-up of the formula, variables are only eliminated if the number of new clauses is less than the number of removed clauses.

- *Pure Predicate Elimination* (PPE) [146] eliminates a predicate symbol P in a formula F if all occurrences of literals with predicate symbol P are of the same polarity.

- *Unused Definition Elimination* (UDE) [146] is a preprocessing method that removes so-called unused predicate definitions from general formulas (i.e., formulas that are not necessarily in CNF).

- *Blocked Clause Elimination* [155] removes blocked clauses from a CNF formula. A blocked clause is redundant in the sense that neither its deletion from nor its addition to F affects the satisfiability or unsatisfiability of F. If a clause C contains a literal L with the pure predicate symbol P, then there are no resolvents of C from L, hence it is vacuously blocked. Therefore, blocked clause elimination removes all clauses that contain pure predicates and thus simulates PPE, and, under some conditions, also UDE.

- *Blocked Clause Decomposition* (BCD) [142] splits a CNF formula into two parts that is then solved via blocked-clause elimination.

- Newer techniques that elimate different variants of covered clauses[14] such as explicit, hidden, and asymmetric clauses are introduced in [140] and found to be more powerful than standard BCE [141].

- HypBinRes, a rule for inferring binary clauses[14] prunes the search space using specialised versions of graph traversal algorithms. However the effect of this preprocessing varies for different problem classes and was found to be best effective for constraint satisfaction problems (CSP) [79], and relatively an ineffective preprocessing algorithm in SAT solvers due to its interaction with the branching heuristic used by the solver.

[14] Given a CNF formula F, a clause $C \in F$ is covered if $R(F, C, l)$ (the resolvent of C w.r.t l) is blocked w.r.t. F [140].

- [139] presents an implementation of clause elimination procedures that are variants of older strategies such as tautology elimination, subsumption elimination, and blocked clause elimination, in MiniSat 2.0 showing significant performance gains, although it is not shown to have tractability of previously intractable ones.

More details on the strengths of different types of clause elimination strategies and their successful implementations can be found at [141, 88, 141, 140, 155, 188]. One of the principal challenges is to achieve a good balance between the time that is spent in preprocessing and the real benefits provided by the simplification. Some clause eliminations techniques are complex and some works have provided improved procedures. For example, identifying blocked clause in polynomial time in [163], and an improved algorithm to identify subsumed clauses in [88].

It is also true that shorter and simpler formulae are not always the ones which are easier to solve [193]. Sometimes having redundant clauses (i.e., clauses that follow as a logical consequence of the rest of the formula) are helpful to more quickly discover conflicts and prune the search space. It is not unusual, in fact, to find instances that become harder after being treated with a preprocessor. Moreover, techniques that "enrich" a formula by adding redundant clauses have been sometimes found useful [192]. *Clause addition procedures*, the dual of clause elimination procedures, add to CNF formulas clauses that are redundant [156]. The most notable examples of these are *Blocked Clause Addition* (BCA) and *clause learning*. [156] reveals that the addition of certain small blocked clauses has shown to be useful when performed in a careful manner. Clause learning is implemented during conflict-analysis to prune the search space and to skip redundant decisions, but algorithms that effectively minimize the number of learned clauses to reduce memory usage and boost solving time is now widely implemented in solvers [254].

Variable elimination with BCE is a simplification technique shown to be very effective in SAT solving - [155, 141, 161].[88] empirically demonstrates the effectiveness of variable elimination, subsumption, subsumption resolution, self-subsumption and definitional subsumption on industrial SAT problems. Their implementation in the SatELite preprocessor which, in combination with the MiniSat solver [90], won all three industrial categories of the SAT 2005 competition [176]. Variable elimination led to a significant redution in number of clauses – upto 74% in the NiVER solver [259]. Many other ideas for formula preprocessing have been proposed in the literature [80, 192, 14, 47], but only a few of them have actually been successful.

In addition to preprocessing, some solvers implement inprocessing rules [34] interleaving simplification and CDCL search or during incremental SAT solving each time a solver is called [95]. Inprocessing using additional deduction rules (Lingeling uses four inference rules LEARN, FORGET, STRENGTHEN, and WEAKEN presented in [156]) was found to improve existing preprocessing techniques such as clause elimination/addition procedures (clause vivification [275, 183], on-the-fly subsumption removal [135, 134, 282]), variable/literal elimination (hidden literal elimination [143], removing redundant literals [28, 254]). There are conflicting reports as well where empirical studies [277] have found inprocessing not as effective as preprocessing. There are other sophisticated prepreprocessing techniques [177, 214, 173], but they do not apply to CNF.

All CNF simplification techniques discussed here work for all SAT problems, independent of whether they have been generated from an original propositional or an FOL problem and independent of the domain. Therefore they can be used in conjunction with optimizations that are FOL - or domain - specific.

3.2.3 Simplification Techniques for FOL Problems

Instantiation-based model finding procedures, specifically the MACE-style method discussed in Section 2.2.2 requires propositional instantiation, which leads to an explosion of variables and clauses. This problem can be alleviated by techniques that specifically deal with FOL problems. Many of the propositional simplification techniques, such as the clause elimination techniques discussed in the previous section are lifted to deal with FOL problem without affecting its satisfiability or unsatisfiability. For example, [162] introduces the principle of implication modulo resolution, which lifts clause-elimination techniques from propositional SAT to FOL. Typical FOL-CNF clause elimination techniques involve the elimination of redundant clauses (tautology, blocked or subsumed), which is undecidable in FOL. Implication modulo resolution provides efficient criteria to identify certain kinds of redundant clauses for elimination. Variable elimination for CNF simplification [33, 88, 259, 75] is generalized as the predicate elimination technique in [161], and implemented in iProver. Predicate elimination is based on two rules: flattening, where all terms are abstracted from P-literals, and flat resolution, where the flattened predicates are resolved. This procedure may or may not lead to the reduction in the number of clauses but will lead to the generation of a different set of clauses. [161] illustrates that iProver with predicate elimination performed extremely well on TPTP problems that standard iProver was unable to solve. iProver's NSR-Pred-Elim algorithm also performs clause simplification based on equality substitution, tautology elimination, subsumption, subsumption resolution and global subsumption. Vampire [172] lifts some propositional redundancy eliminations techniques for FOL through an improved clausification algorithm [146, 161], and also implements a generalization of blocked-clause elimination as a preprocessing step [163]. Experimental results proved that blocked-clause elimination helped Vampire, iProver and CVC4 solve new satisfiable problems that previously could not be solved [163]. However, we show that we can achieve better performance with Vampire and, sometimes, with iProver using our proposed definition elimination technique (results presented in Chapter 7).

Alternatively, the instantiation-based procedure implemented in Darwin prover, the model evolution calculus [24] simplifies an FOL formula to function-free clause logic (not SAT) which leads to an almost linear increase in size of formula wrt to domain size unlike propositional logic which is exponential. However based on experimental results presented in [23], the average time used by Darwin to solve satisfiable problems from the TPTP problem set was at least 35% greater than Paradox's runtime, although FM-Darwin claims to scale much better with larger domain sizes compared to Paradox. It is clear from CADE-ATP competition results that solvers are very dependent on the kind of problems – Paradox beat FM-Darwin in the SAT division whereas FM-Darwin performed really well in the EPR division[15] [212, 23]. The nature of the FOL ontologies is not well aligned with EPR and thus Darwin is not expected to perform well on our ontologies.

Other heuristic techniques exploited in incremental SAT model finding tools include (1) re-using pre-constructed interpretations as initial values to improve MACE-style finite model finding in Paradox [56], (2) identifying and removing unnecessary axioms for a specific reasoning task [46], (3) symmetry breaking in CNF graphs to prune the search space [113], (4) non-ground splitting [240] (implemented in the program `eground` and adopted in E – an instantiation based prover) to reduce number of variables in a clause – this applies only to near-propositional CNF formulae, whose signature does not have any function symbols, (5) principled addition of redundancy to formulas for efficient grounding algorithms [276]. Some other techniques specific to theorem proving include (1) pseudo-splitting for saturation-based theorem proving in Vampire [233], (2) contraction techniques such as generalization inference rule to discard or simplify instances [211, 276] specific for resolution on EPR formulas, (3) using answer set programming (ASP) such as `SATGRND` [111].

Except simplification proposed in [46] to minimise the amount of information given to the model builder, the rest of the discussed preprocessing techniques are either applied to a propositional logic, or CNF representations of FOL problems and not on the FOL problem itself. Some of the complexity of preprocessing can be reduced if simplification can be performed much cheaply at the FOL level before its translation to FOL-CNF or propositional-CNF when the size of the problem increases polynomially or exponentially. Optional definition elimination that we introduce in Chapter 5 allows to us to simplify the FOL problem directly, which can then be still subjected to any of the preprocessing or inprocessing techniques that work on the CNF and SAT representations.

[15] SAT division contains problems in propositional logic; EPR division contains problems in effectively propositional logical also called the Bernays-Schönfinkel-Ramsey fragment of FOL, where the problem contains no function symbols

3.3 Reasoning with Spatial Ontologies

This section presents a review of spatial representations for managing and reasoning of spatial information in Geographic Information Systems (GIS), and spatial ontologies that support automated reasoning about the semantics of spatial information, in particular with a combination of qualitative and geometric information.

3.3.1 Spatial Ontologies in Geospatial Ontology Standards

While many spatial ontologies have been developed [252, 121, 127], only few of them actually axiomatize spatial semantics to a degree that is sufficient to support automated reasoning with and not just querying of spatial information. We briefly introduce some standards in which most spatial data are represented or stored (they only provide a high-level conceptual framework of spatial concepts with minimal semantics, axiomatic relations or ontological commitments) and then highlight upon a few comprehensive axiomatic qualitative ontologies – and this guides the selection of ontology we want to integrate the standards with for an integrated reasoning.

Foundational ontologies: SUMO [213], DOLCE [198], the BFO-SNAP ontology [121] and GFO [19] either contain too few, only high-level spatial concepts and relations to support any specific spatial reasoning, or the relations are not at all or only sparsely axiomatized. In addition, without a mapping to geometric ontologies reasoning with the available geometric data is impossible.

Geospatial domain ontologies: Most geospatial ontologies only represent geometric objects, such as points (the classical representation for location, making use of the latitude and longitude properties defined in RDF in the W3C Geo vocabulary[16]), regions and curves (represented by a collection of points such as in OpenGIS standard used in LinkedGeoData [9] and GeoLinkedData [18]), allowing access to only geometric data, but include no qualitative spatial relations. W3C Geo is a widely used vocabulary for geometric objects, and Ordnance

[16]`http://www.w3.org/2003/01/geo/wgs84_pos`

Survey (OS)[17] for spatial relations. Ordnance Survey Spatial Relations Ontology[18] includes topological operators, in addition to properties for describing metric location (easting and northing), while the NeoGeo spatial ontology is restricted to topological relations, but neither ontologies axiomatize them and thus does not afford the capability of reasoning over pure qualitative information or even extract qualitative information from geometric data. Some standards, like GeoSPARQL[19] defines top-level RDFS/OWL classes for geometric object types from OpenGIS Simple Features – the standard that we will formalize using CODI in Chapter 4, and includes mereotopological relations from the 9-intersection, but only for querying geometric datasets. They do not include an axiomatization of these relations that support qualitative reasoning. [257] presents an ontology of 0-2 dimensional geometric configurations. Its relations pertaining to topology (RCC5), distance (LNF) [81], orientation [58, 86], direction relations [119], adjacency (wordnet), collocation and object parthood are made available for SQL querying – although no axiomatization is available. Moreover any kind of qualitative reasoning available using these standards relies on underlying geometric data for inferencing rather does not allow pure qualitative spatial reasoning. And pure FOL-based extensively axiomatized qualitative ontologies (e.g., RCC or CODI by itself) do not support using geometric data for reasoning.

Systems (such as Ontop-spatial [30]) have been designed to answer queries on top of geospatial data that reside in RDF stores, such as Parliment, uSeekM, Virtuoso, Stardog etc. Querying in these cases is offered by OGC standards such as GeoSPARQL [220], stRDF, stSPARQL [178]. But as highlighted, these basic standards are still essentially taxonomies and provide a hierarchy of geometric objects such points, lines and areas and a set of spatial relations but contains no semantic formalization between entities and relation. They therefore offers very minimal support in terms of any kind of advanced spatial reasoning beyond the extraction of subclass-hierarchy.

[17] `http://data.ordnancesurvey.co.uk/ontology/spatialrelations`
[18] `http://data.ordnancesurvey.co.uk/ontology/spatialrelations/`
[19] `http://www.opengeospatial.org/standards/geosparql`

3.3.2 Integrated Qualitative and Quantitative Spatial Reasoning

Simple mereotopological relations included in popular geospatial data standards used in GIS systems such as OGC Simple Features employed in ArcGIS mostly use the 9-intersection method [91, 92], its dimension-extended refinement (DE-9I) [57] and extensions thereof [60, 202, 238]. These standards determine qualitative spatial relations from an underlying geometric representation with associated operations for determining their boundary and interior, for all involved objects. Moreover, the semantics of the mereotopological relations, especially their interaction (e.g., parthood specializes overlap or a whole is in contact with everything any of its parts is on contact with), are never explicitly captured (e.g., as axioms) and thus not available for qualitative reasoning with the underlying data. And therefore these relations cannot be used for reasoning where geometric data models are not the only source of qualitative information. This is in sharp contrast with axiomatic treatments of mereotopology such as the RCC [226], which axiomatically constrain the interpretations of qualitative spatial relations, such as contact and/or parthood, and define other relations, such as overlap or external contact [52]. By explicitly formalizing relationships between the relations, axiomatic frameworks permit spatial reasoning with qualitative information even in the absence of geometric information. However, axiomatic theories of mereotopology have, in the philosophical tradition of Whitehead, been often married to strict region-based conceptualizations of space wherein extended spatial entities – typically called regions – are the only first-class entities of the domain, while points and other lower-dimensional entities are not entities in the domain [127]. A hybrid reasoning system utilizing a constraint network reasoning approach for reasoning with both geometric and qualitative information has been presented in [96]. Our work in Chapter 4 goes a step further by explicitly formalizing the semantics relationships between the two types of information for reuse with any logic-based reasoner. We accomplish this by taking a qualitative axiomatic theory and connecting it to geometric data models in order to permit joint qualitative-geometric reasoning. But the limitation to make this realization is the use of regions of only one dimensionality in traditional axiomatic theories which make them incompatible with the geometric data models. This prevents full integration with geometric data standards, such as Simple Features, that permit entities of different dimensions. The idea of *multidimensional mereotopology* [107, 118, 130, 251] aims to overcome this restriction by axiomatically formalizing mereotopological relations not just between entities of equal dimensions but also between entities of different dimensions. CODIB [130, 128, 127] as one such multidimensional axiomatic theory allows entities of different dimensions to coexist similar to how such geometries are used in spatial data standards, and therefore can be used to qualitatively generalize geometric data models. To enable the kind of joint and stand-alone qualitative and quantitative spatial reasoning that we aim to achieve, we therefore use CODIB as the foundational framework for formalizing Simple Feature Access schema's semantics in Chapter 4. Then, we use geometric data in conjunction with this combined qualitative-geometric ontology to test external ontology verification as one particular kind of hybrid qualitative-geometric reasoning task.

Qualitative spatial calculi (see the overview in [64]) are yet another approach to qualitative spatial reasoning, but they can only incorporate qualitative information and cannot make use of geometric information without first translating it to qualitative information, and thus is also incapable to achieve the kind of integrated reasoning we aim for.

CHAPTER 4

FORMAL QUALITATIVE SPATIAL AUGMENTATION OF THE SIMPLE FEATURE ACCESS MODEL

The need to share and integrate the large amounts of heterogeneous geospatial data has resulted in the development of geospatial data standards, such as OGC's GeoSPARQL [220], and the shared OGC/ISO standards Geography Markup Language (GML) [152] and Simple Feature Access [151]. All of these standards include some types of simple and complex geometric features – often simply referred to as *geometries* – for representing geographic objects. The most commonly used features include points, line segments and aggregations into polylines, and polygons and aggregations into polyhedral surfaces. Primarily concerned with interoperability across spatial databases and geographic information systems, these standards also prescribe a number of common spatial operators, e.g. for calculating intersections, differences, buffers, or distances between features. Many of these standards have further incorporated a number of simple mereotopological relations (with Boolean values), such as intersects, contains, overlaps, meets, or crosses. These are based on results from the Region Connection Calculus (RCC) [226] and the almost equivalent topological relations defined by the 9-intersection method [91, 92] and its dimensionally extended refinements (DE-9I) [57, 60] and further extensions [202, 238]. However, these relations are provided as query operators only, allowing one to access geometric data in a more natural way[1] . But without formalizing the relationships between geometric representations and qualitative relations, these approaches cannot support qualitative reasoning over the queried information. Moreover, storing "native" topological information – for example as provided from textual sources where precise locations or spatial extents are unknown or unknowable – is currently not possible without having to *invent geometric objects*. For example, the spatial content of the two statements "Lot A is for sale and abuts Broadway." and "Lot B that does not border

[1]Most GIS support the RCC or DE-9I relations, with recent progress on storing the computed relations more efficiently [187]. There has also been a call to extend this to a larger set of qualitative relations [99].

Broadway is not for sale." cannot be represented in GIS without assigning geometries to the named objects.

Frameworks for qualitative spatial representation and reasoning (see, e.g. the overview in [64]) such as the RCC support direct reasoning about topological and other kinds of qualitative spatial information (e.g. direction), but cannot easily mix geometric data sources (e.g. the precise location of "Broadway") and qualitative information (the fact that "Lot A" and Broadway are connected) to infer which lots on a property map may be for sale. Similar interpretation of qualitative spatial information on a geometric dataset is needed during natural disasters, when interpreting human reports (e.g. from social media or news reports) on road networks, elevation data, and hydrological data, to help answer simple queries, such as "is any part of the historic center flooded?".

Towards objective 1 (O1 in Section 1.2.2) of this dissertation, we develop a first-order logical ontology that treats geometric features (e.g. polylines, polygons) and relations between them as specializations of more general types of features (e.g. any kind of 2D regions or 1D features) and mereotopological relations between them. Key to this endeavour is the use of a *multidimensional* theory of space wherein, unlike traditional logical theories of mereotopology (including the RCC), spatial entities of different dimensions can co-exist and be related. We choose the theory CODIB (based on CODI [130, 128] with an extension by boundary/interior distinctions [127]) as the suitable multidimensional theory of qualitative space and test to what extent geometric features from SFA [151] can be treated as specializations of CODIB's more general non-geometric spatial feature types from CODIB. For example, SFA's line segments or polylines should specialize the general one-dimensional spatial features, called "curves", from CODIB. Specifically, we want to leverage the detailed formal semantics encoded in CODIB to capture the semantics of SFA's various geometric feature types and mereotopological relations in greater detail. Currently, much of these semantics are described in natural language and mathematical notation in the standard, but are not accessible to automated reasoning. Wherever possible, we logically define SFA's geometric features in terms of

CODIB's spatial concepts and, where that is not possible, treat them as specializations with suitable constraints.

Our specific contributions are: (1) developing a first-order logic axiomatization, called SF-FOL, of SFA; (2) in the process, show that all of the geometric feature types from SFA specialize or map to types of spatial entities definable in CODIB; (3) fully define SFA's mereotopological relations in CODIB and thus provide computer-interpretable semantics of these qualitative relations; and (4) verify the consistency of SF-FOL. This makes both SFA's and CODIB's mereotopological relations applicable to geometric and qualitative data alike and allows using automated first-order logic theorem provers (ATPs) for integrated mereotopological reasoning over combinations of qualitative and geometric data from any sources that adhere to the SFA standard.

4.1 Preliminaries

We now review and formalize the relevant aspects of the SFA standard, namely its classes of geometric features and its qualitative relations. In particular, Section 4.1.1 formalizes the intrinsic semantics of the UML subclass hierarchy from the standards document in first-order logic as a starting point for its semantic enhancement. Subsequently, Section 4.1.2 reviews key relations and concepts from the CODI and CODIB ontologies and provides definitions of novel concepts that are necessary to draw some of the distinctions that SFA makes. These concepts and relations will be used as basis for elaborating the SFA semantics and making its geometric features available for integration with purely qualitative information and for general qualitative reasoning.

All logical sentences throughout our exposition are assumed to be universally quantified. They are labeled in the format '[ontology]-[type][number]' (e.g. SFC-T1) where the first letter(s) indicate the ontology (e.g. SFC=simple features concept, SFR=simple features relation, PO=partial overlap, D=dimension), while the type distinguishes axioms (A), definitions (D: defining a concept or relation), theorems (T: a property provable from the axioms and definitions), and mappings (M: an axiom that establishes some relationship between SFA and CODIB). All axioms, definitions and theorems for SF-FOL are available in modularized form in the Common Logic syntax from the COLORE repository[2] .

4.1.1 Semantics of Simple Feature Concepts and Spatial Relations

Simple Features Access (SFA) [138], is an OGC and ISO standard for vector-based encoding of 0-2D geometric data that aims to facilitate interoperability across GIS and spatial databases. For example, SFA is at least partially implemented by ArcGIS, PostGIS, and the spatial extensions of MySQL, Oracle, and IBM Db2. Other standards, like GeoSPARQL [220] and GeoJSON, build on it.

4.1.1.1 Semantics of Concepts (Classes) from Simple Features

At the core of the SFA lies a set of simple geometries – called simple features – such as individual points (*sf_point*), polylines (*sf_line_string*: a sequence of straight line segments), and polyhedral surfaces (*sf_polyhedral_surface*: a connected, possibly non-planar 2D area obtained by stitching polygons together). *Sf_line_string* and *sf_polyhedral_surface* specialize the abstract, non-instantiable classes *sf_curve* (which may include non-straight segments) and *sf_surface* (which may include 2D areas with non-straight boundary segments), respectively (SFC-A1,A2), that capture 1D and 2D spatial objects more generally[3] . In addition to the three classes of simple features, collections of simple features can be modeled using the *sf_geometry_collection* class. All four specializations of the abstract class *sf_geometry* are mutually disjoint (SFC-A3-A6) and jointly exhaustive (SFC-D1).

(SFC-D1) $\mathit{sf\_geometry}(x) \leftrightarrow \mathit{sf\_point}(x) \vee \mathit{sf\_curve}(x) \vee \mathit{sf\_surface}(x) \vee \mathit{sf\_geometry\_collection}(x)$

(SFC-A1) $\mathit{sf\_line\_string}(x) \rightarrow \mathit{sf\_curve}(x)$

[2]In https://colore.oor.net/. Note that all of axioms are specified using only the classical first-order logic syntax of Common Logic and without use of any of Common Logic's specialized features such as restricted module import or use of sequence markers. This allows easy translation to pure first-order logic representations such as the TPTP format [264] supported by many theorem provers and model finders.

[3]Throughout our formalization, axioms are always assumed to be universally quantified.

(SFC-A2) $sf\_polyhedral\_surface(x) \rightarrow sf\_surface(x)$

(SFC-A43) $sf\_point(x) \rightarrow \neg sf\_curve(x) \wedge \neg sf\_surface(x) \wedge \neg sf\_geometry\_collection(x)$

(SFC-A4) $sf\_curve(x) \rightarrow \neg sf\_point(x) \wedge \neg sf\_surface(x) \wedge \neg sf\_geometry\_collection(x)$

(SFC-A5) $sf\_surface(x) \rightarrow \neg sf\_point(x) \wedge \neg sf\_curve(x) \wedge \neg sf\_geometry\_collection(x)$

(SFC-A6) $sf\_geometry\_collection(x) \rightarrow \neg sf\_point(x) \wedge \neg sf\_curve(x) \wedge \neg sf\_surface(x)$

Sf_line_string is further specialized by *sf_line* (SFC-A7), which represents a single straight line segment, and *sf_linear_ring* (SFC-A9), a linear feature that is closed, that is, its start and end points are identical and thus its boundary is empty. Note that while we review here the intended semantics of these concepts, we – for now – formalize only what can be expressed using SFA's terminology. The intended semantics are more fully formalized by the mapping to CODIB concepts developed in Section 4.2.1. For example, SFC-M3, M4, M8, and M9 together with CODIB's formalization (including the definitions AtomicS-D, SimpleS-D, BranchedS-D, ConS-D and the formalization of the predicate ICon from [127]) entail that any *sf_line* is a connected curve with two distinct end points. Likewise, *sf_polygon* is a specialization of *sf_polyhedral_surface* (SFC-A9), capturing a planar 2D area with a single closed polyline as exterior boundary[4] . Another specialization of *sf_polyhedral_surface* is *sf_tin* (SFC-A10), a triangulated irregular network (TIN), which should only consist of triangles. A single triangle, described by *sf_triangle*, is a polygon and the simplest kind of TIN (SFC-D2). It must be bounded by a closed polyline (i.e. a *sf_linear_ring*) that consists of exactly three line segments (i.e. *sf_line*), which will be formalized by SFC-M13 in Section 4.2.1.

(SFC-A7) $sf\_line(x) \rightarrow sf\_line\_string(x)$

(SFC-A8) $sf\_linear\_ring(x) \rightarrow sf\_line\_string(x)$

(SFC-A9) $sf\_polygon(x) \rightarrow sf\_polyhedral\_surface(x)$

[4]SFA models *sf_polygon* and *sf_polyhedral_surface* as separate specializations of *sf_surface*, but permits polyhedral surfaces to consist of a single polygon, in which case it is spatially a polygon.

(SFC-A10) $sf\_tin(x) \rightarrow sf\_polyhedral\_surface(x)$

(SFC-D2) $sf\_triangle(x) \leftrightarrow sf\_polygon(x) \wedge sf\_tin(x)$

Sf_multi_point, *sf_multi_curve* and *sf_multi_surface* are special types of *sf_geometry_collection*s (SFC-A11) that are aggregations of only *sf_point*s, *sf_curve*s, or *sf_surface*s, respectively. *Sf_multi_curve* and *sf_multi_surface* are again abstract classes in SFA, with only the specializations *sf_multi_line_string* (SFC-A12) and *sf_multi_polygon* (SFC-A13) being instantiable. The latter two consist only of *sf_line_string*s and *sf_polygon*s, respectively – cf. Section 4.2.2.

(SFC-A11) $sf\_multi\_point(x) \vee sf\_multi\_curve(x) \vee sf\_multi\_surface(x) \rightarrow sf\_geometry\_collection(x)$

(SFC-A12) $sf\_multi\_line\_string(x) \rightarrow sf\_multi\_curve(x)$

(SFC-A13) $sf\_multi\_polygon(x) \rightarrow sf\_multi\_surface(x)$

4.1.1.2 Spatial Relations in Simple Features

In addition to many geometric/quantitative spatial operations (e.g. buffer, intersection, convexHull), which are only well-defined on geometric features (e.g. polygons rather than general surfaces), SFA includes eight named qualitative spatial relations based on the dimension-extended 9-intersection method [57] that equally apply to generalizations of geometric features such as general curves and surfaces. These include the five primitive relations *disjoint*, *touches*, *within*, *overlaps*, and *crosses*. Three additional relations *contains* (inverse of *within*), *intersects* (negation of *disjoint*), and *equals* (conjunction of *within* and *contains*) are defined. These are defined in terms of the interior, boundary, and exterior of the objects in question as documented in the SFA standard [151]. Three dimensional constraints are explicitly mentioned in SFA: touches does not apply to points (or *sf_multi_point*s), overlaps requires the involved entities to be of equal dimension, and crosses is not applicable to two surfaces (or *sf_multi_surface*s). Later, we show that these constraints are provable as theorems of our CODI-based formalization of these spatial relations.

4.1.2 Dimensional Features and Qualitative Spatial Relations in CODIB

This work utilizes the multidimensional mereotopology CODIB [130, 128, 127], which has been specifically developed to qualitatively generalize geometric data models, as basis for formalizing SFA's semantics. This subsection reviews CODIB, whose core is CODI (already reviewed in Section 2.4.2.1 in Chapter 2), and then the additional relation of boundary containment. A computer-readable encoding of the axioms are provided in the Common Logic syntax in the COLORE repository[5] to facilitate automated verification and reasoning.

4.1.2.1 CODI

Core to CODIB is the theory CODI of containment - $\mathit{Cont}(x, y)$, and relative dimension - $\leq_{\dim}(x, y)$. The relations *Cont* and C (C(x, y) - where x and y share a contained object) in CODI are the qualitative generalization of SFA's contains and intersect relations. While CODI does not distinguish different primitive types of entities, they can be defined: *PointRegion*s (which encompass individual points *Point* and sets of points) are of minimal dimension, *Curve*s are of next higher dimension, and so forth [129]. These primitive classes are a specialization of spatial region S from [198], which represents abstract nonzero space occupied by any physical object. One further pertinent classification of spatial entities is based on *internal connectedness* (ICon-D), which requires each proper part y to be connected to its complement $x - y$ such that the shared entity (denoted by the intersection of y and $x - y$) is of exactly one dimension lower than x[6]. For example, two polygons that share a line segment as boundary are internally connected, but if they only share a point, they are not.

(PR-D) $\mathit{PointRegion}(x) \leftrightarrow S(x) \wedge \mathit{MinDim}(x) \wedge \neg \mathit{ZEX}(x)$ ***(point sets)***

(Point-D) $\mathit{Point}(x) \leftrightarrow S(x) \wedge \mathit{Min}(x) \wedge \mathit{MinDim}(x)$ ***(points)***

(Curve-D) $\mathit{Curve}(x) \leftrightarrow S(x) \wedge \forall y[\mathit{PointRegion}(y) \rightarrow y \prec_{\dim} x]$ ***(curves (1D entities))***

(AR-D) $\mathit{ArealRegion}(x) \leftrightarrow S(x) \wedge \forall y[\mathit{Curve}(y) \rightarrow y \prec_{\dim} x]$ ***(areal regions (2D entities))***

[5] Various strengths of the theories can be found at `colore.oor.net/multidim_mereotopology_codi` and `colore.oor.net/multidim_mereotopology_codib`

[6] See [128] for the full axiomatization of the intersection and complement operations in CODI.

(ICon-D) $ICon(x) \leftrightarrow \forall y[PP(y,x) \rightarrow C(y, x-y) \wedge y \cdot (x-y) \prec_{\text{dim}} x]$ ***(internally connected)***

4.1.2.2 CODIB

CODIB is a logical extension of the the theory CODI, meaning that is adds additional axioms. Most importantly, CODIB utilizes an additional primitive relation of boundary containment, $BCont(x,y)$. *BCont* specializes containment and incidence (BC-A1) and is irreflexive, asymmetric and transitive with respect to containment. While a boundary-contained entity must be of a lower dimension than the containing entity, it is not necessarily of the next-lower dimension. For example, an areal (i.e. 2D) region can contain both curves and points in its boundary. Note that *BCont* is a primitive because it cannot be defined in CODI, meaning that in some models of CODI it cannot be determined whether a contained entity is actually contained in the boundary or interior of some containee.

(BC-A1) $BCont(x,y) \rightarrow Cont(x,y) \wedge Inc(x,y)$

4.1.2.3 Refined Spatial Region Concepts in CODIB

CODIB refines spatial regions based on whether and how their parts are connected, resulting in the subclass hierarchy of spatial regions with different properties that is shown in Figure 4.1. A connected region (ConS-D) is *internally-connected*, while its complement is a multipart region (MS-D). A simple region has proper parts that are connected but are non-branched (Simple-D). A connected region that contains at least three non-overlapping proper parts that share an entity of lower dimension is called a branched region (BranchedS-D). An atomic region is a simple region without any proper parts (Atomic-D).

(ICon-D) $ICon(x) \leftrightarrow \forall y[PP(y,x) \rightarrow C(y, x-y) \wedge y \cdot (x-y) \prec_{\text{dim}} x]$ ***(internally connected)***

(ConS-D) $Connected\_S(x) \leftrightarrow S(x) \wedge ICon(x)$ ***(connected spatial region)***

(MS-D) $Multipart\_S(x) \leftrightarrow S(x) \wedge \neg Connected\_S(x)$ ***(multipart spatial region)***

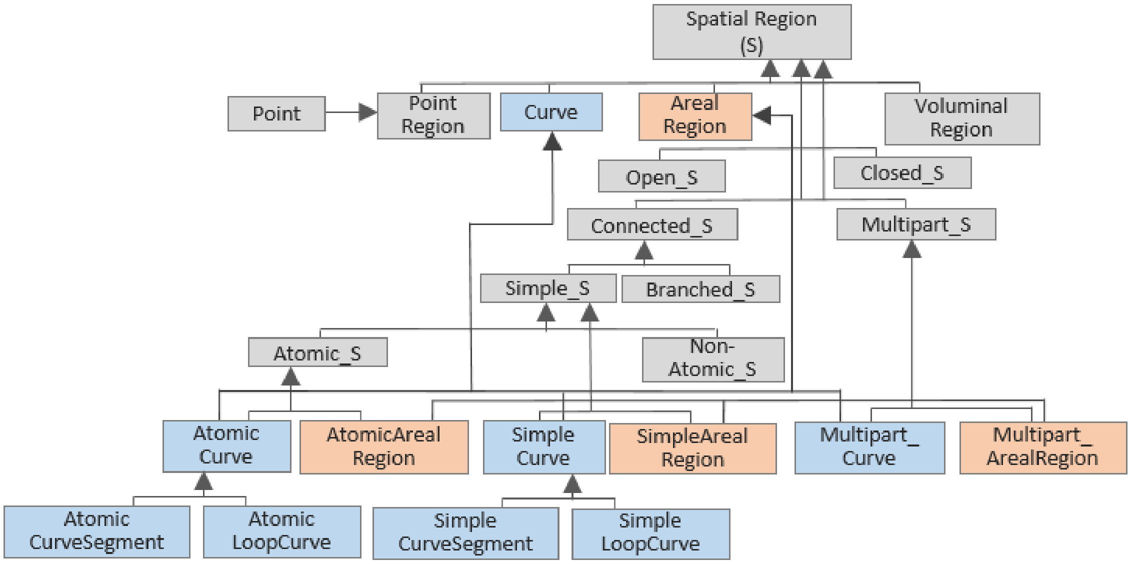


Figure 4.1: Taxonomy of refined CODIB spatial region concepts classified based on presence/absence of boundaries, connectedness, branching and parts

(BranchedS-D) $Branched\_S(x) \leftrightarrow Connected\_S(x) \wedge \exists p, q, r, s[PP(p, x) \wedge PP(q, x) \wedge PP(r, x) \wedge \neg PO(p, q) \wedge \neg PO(p, r) \wedge \neg PO(q, r) \wedge s \prec_{\text{dim}} p \wedge s \prec_{\text{dim}} q \wedge s \prec_{\text{dim}} r \wedge Cont(s, p) \wedge Cont(s, q) \wedge Cont(s, r)]$ **(*A branched spatial region is a connected region that has three distinct non-overlapping parts* p, q, r *that all share a common lower-dimensional entity* s*. For example, a branched curve has three non-overlapping segments that all share a point.*)**

(SimpleS-D) $Simple\_S(x) \leftrightarrow Connected\_S(x) \wedge \neg Branched\_S(x)$ **(*simple spatial region*)**

(AtomicS-D) $Atomic\_S(x) \leftrightarrow Simple\_S(x) \wedge Min(x)$ **(*an atomic spatial region is a simple spatial region that is minimal, i.e. has no proper parts*)**

These properties are now used to define specialized classes of curves and areal regions.

(SCS-D) $SimpleCurveSegment(x) \leftrightarrow Curve(x) \wedge Simple\_S(x) \wedge \exists p, q[BCont(p, x) \wedge BCont(q, x) \wedge p \neq q]$ **(*Simple curve segment has two distinct end points*)**

(SLC-D) $SimpleLoopCurve(x) \leftrightarrow Curve(x) \wedge Simple\_S(x) \wedge \forall y[Point(y) \rightarrow \neg BCont(y, x)]$

(*Simple loop curve is closed: it does not contain any point in its boundary*)

(ACS-D) $AtomicCurveSegment(x) \leftrightarrow SimpleCurveSegment(x) \wedge Atomic\_S(x)$

(ALC-D) $AtomicLoopCurve(x) \leftrightarrow SimpleLoopCurve(x) \wedge Atomic\_S(x)$

(SAR-D) $SimpleArealRegion(x) \leftrightarrow ArealRegion(x) \wedge Simple\_S(x)$

(MC-D) $Multipart\_Curve(x) \leftrightarrow Curve(x) \wedge Multipart\_S(x)$

(MAR-D) $Multipart\_ArealRegion(x) \leftrightarrow ArealRegion(x) \wedge Multipart\_S(x)$

4.2 Axiomatization of Simple Feature as an Extension of CODIB

In this section we present the core of our formalization that elaborates the semantics of the concepts in the skeleton axiomatization of SFA from Section 4.1.1 using qualitative concepts and relations from CODI(B). This results in two new ontologies that logically extend SFC-Core and CODIB: SFC-FOL, which includes the more detailed axiomatization of SFA's concepts, and SFR-FOL, which axiomatizes SFA's mereotopological relations. Figure 4.2 summarizes the taxonomic relationships between the SFA and CODI(B) concepts, but the real contribution are the detailed axiomatic mappings.

4.2.1 Axiomatization of Simple Feature's Simple Geometric Features

The base geometry class *sf_geometry* is a specialization of spatial region *S* (SFC-M1) from [198]. The elementary geometry classes *sf_point*, *sf_curve*, *sf_surface*, and *sf_geometry_collection* are disjoint and exhaustive subclasses of *sf_geometry*. *Sf_point* and *sf_surface* are specializations of CODI's *Point* and *ArealRegion* (SFC-M2,C6) respectively. CODI's *Curve* is a generalization of curves that are open, closed and infinite, whereas *sf_curve* only includes simple curve segments and loop curves (SFC-M3). Since the description for *sf_curve* requires additional axioms to constrain its meaning, SFC-M3 is an axiom (using implication instead of bi-conditional) rather than a definition. A *sf_curve* that is a *SimpleCurveSegment* has a start and end point that are distinct (SFC-M4). A *sf_curve* that is a *SimpleLoopCurve* has start and end points that are identical (SFC-M5). It also does not contain any point in its boundary

(SFC-T1). SFA's definition of curve rules out branching curves. *Sf_geometry_collection* is either a multipart or branched spatial region that places no constraints on its elementary geometric parts. Subclasses of *sf_geometry_collection* have restricted membership (it only allows parts of identical dimension) with additional constraints on the degree of spatial overlap between individual elements. The axioms SFC-M1 to C7 suffice to tie in most simple geometric features to the qualitative ontology CODI and CODIB to perform simple consistency checking and mereotopological reasoning over simple geometric features.

(SFC-M1) $sf\_geometry(x) \leftrightarrow S(x)$ (***sf_geometry is equivalent to DOLCE's Spatial Region***)

(SFC-M2) $sf\_point(x) \leftrightarrow Point(x)$ (***sf_point is equivalent to CODI Point***)

(SFC-M3) $sf\_curve(x) \rightarrow SimpleCurveSegment(x) \vee SimpleLoopCurve$ (***sf_curve is either CODIB's SimpleCurveSegment or SimpleLoopCurve***)

(SFC-M4) $sf\_curve(x) \wedge SimpleCurveSegment(x) \rightarrow \exists p1, p2[sf\_point(p1) \wedge sf\_point(p2) \wedge sf\_start\_point(p1, x) \wedge sf\_end\_point(p2, x) \wedge BCont(p1, x) \wedge BCont(p2, x) \wedge p1 \neq p2]$

(***A sf_curve that is a curve segment has distinct start and end points that are boundary contained***)

(SFC-M5) $sf\_curve(x) \wedge SimpleLoopCurve(x) \rightarrow [\exists p1, p2[sf\_point(p1) \wedge sf\_point(p2) \wedge sf\_start\_point(y, x) \wedge sf\_end\_point(z, x)]] \rightarrow y = z$ (***A sf_curve that is a loop curve has the same start and end point***)

(SFC-T1) $sf\_curve(x) \wedge SimpleLoopCurve(x) \rightarrow \neg\exists y[sf\_point(y) \wedge BCont(y, x)]$

(***A*** *sf_curve* ***that is a loop curve does not contain any point in its boundary***)

(SFC-M6) $sf\_surface(x) \leftrightarrow ArealRegion(x)$ (***sf_surface is equivalent to CODI ArealRegion***)

(SFC-M7) $sf\_geometry\_collection(x) \rightarrow Multipart\_S(x) \vee Branched\_S(x)$

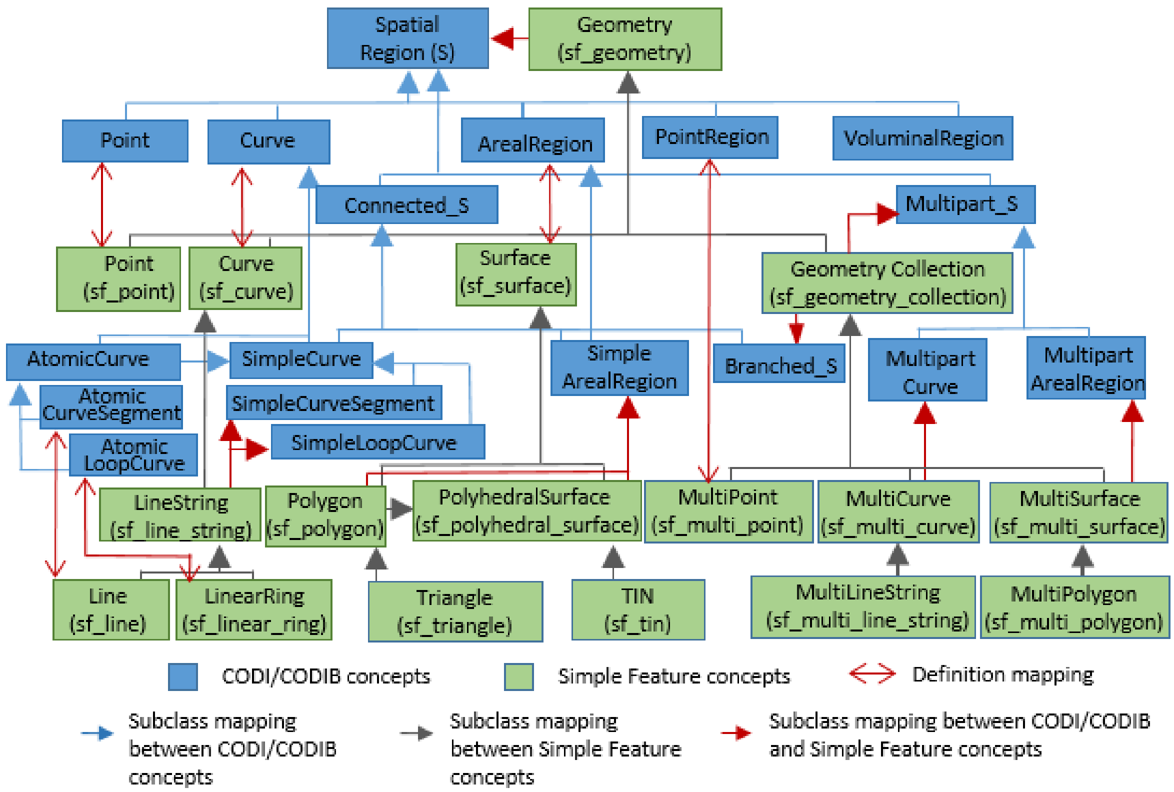


Figure 4.2: Hierarchy of SF-FOL indicating mapping within SFA concepts, within CODI/CODIB concepts and between SFA and CODI/CODIB concepts.

(***sf_geometry_collection is a specialization of either CODIB's multipart or a branched spatial region***)

At the secondary level the notions of *connectedness*, *open/closed* and *atomic/simple* (non-atomic)/ *branched* are used to distinguish more refined geometric concepts. *Curve* in CODIB is (a) atomic if it has exactly one start point and one end point, and (b) closed when its two end points are be identical. *Sf_line* is an *AtomicCurveSegment* that has exactly 2 points (SFC-M9) contained in its boundary. The boundary of a topologically closed *Curve* is empty, which means its start point is the same as its end point and this point is not boundary-contained ($\neg BCont$). *Sf_linear_ring* is both an atomic and closed curve (SFC-M10). *Sf_line_string* is a simple curve with linear interpolation between points (SFC-M8) with minimal parts that are *AtomicCurveSegments*. We can infer that *sf_line_string* generalizes *sf_line* and *sf_linear_ring* (SFC-A8,A9) as theorems (from SFC-M8-C10).

(SFC-M8) $sf\_line\_string(x) \leftrightarrow sf\_curve(x) \wedge \forall y[PP(y,x) \wedge Min(y) \rightarrow AtomicCurveSegment(y)]$

(***sf_line_string is an sf_curve whose minimal parts are CODIB's AtomicCurveSegments***)

(SFC-M9) $sf\_line(x) \leftrightarrow AtomicCurveSegment(x)$ (***sf_line is equivalent to CODIB AtomicCurveSegment***)

(SFC-M10) $sf\_linear\_ring(x) \leftrightarrow sf\_line\_string \wedge AtomicLoopCurve(x)$ (***sf_linear_ring is equivalent to sf_line_string and AtomicLoopCurve***)

Sf_surface is a 2-dimensional geometric object that may be an atomic (associated with one 'exterior boundary') or a simple (non-branching) areal region. *Sf_polygon* is a simple areal region (SFC-M11), and each interior boundary defines a hole in the polygon. The boundary of a *sf_surface* is the set of closed curves (*sf_linear_ring*s) that make up its exterior and interior boundaries (SFC-T1). A *sf_polyhedral_surface* is a simple areal region formed by 'stitching' together *sf_polygon*s along their common boundaries (SFC-M12). Such surfaces in a 3-dimensional space may not be planar as a whole, depending on the orientation of their planar normals. If all the polygons are in alignment (their normals are parallel), then the whole stitched polyhedral surface is co-planar and can be represented as a single polygon if it is connected. If a *sf_polyhedral_surface* is closed, then it bounds a solid. No two rings in the boundary of a *sf_surface* cross and the rings in the boundary of a polygon may intersect at a point but only as a tangent. A *sf_triangle* is a *sf_polygon* (SFC-M13) with 3 distinct, non-collinear vertices and no interior boundary. The exterior boundary defines the 'top' of the surface which is the side of the surface from which the exterior boundary appears to traverse the boundary in a counter clockwise direction. The interior boundary will have the opposite orientation, and appear as clockwise when viewed from the 'top'. *Sf_tin* is a *sf_polyhedral_surface* whose minimal parts are *sf_triangle*s (SFC-M14) .

(SFC-M11) $sf\_polygon(x) \rightarrow SimpleArealRegion(x) \wedge \exists y,z[BCont(y,x) \wedge ICon(y) \wedge Closed(y) \wedge boundary(z) = y \wedge P(x,z)]$ (***sf_polygon specializes CODIB's***

SimpleArealRegion and some part y of its boundary – the exterior boundary – is internally connected and closed and bounds a region z of which x is part. This construct is necessary to accommodate polygons with holes bounded by parts of their boundary. For polygons without holes z=x can be chosen, and then z is the entire boundary of x.)

(SFC-T2) $sf\_polygon(x) \wedge BCont(y, x) \rightarrow sf\_linear\_ring(y)$ (***The boundary of sf_polygon is a sf_linear_ring***)

(SFC-M12) $sf\_polyhedral\_surface(x) \leftrightarrow SimpleArealRegion(x) \wedge ICon(x) \wedge \forall y[P(y, x) \wedge Min(y) \rightarrow sf\_polygon(y)]$ (***sf_polyhedral_surface is equivalent to CODIBs SimpleArealRegion that is internally-connected and is an aggregation of sf_polygons***)

(SFC-M13) $sf\_triangle(x) \leftrightarrow sf\_polygon \wedge \exists p, q, r[p \neq q \wedge p \neq r \wedge q \neq r \wedge sf\_line(p) \wedge sf\_line(q) \wedge sf\_line(r) \wedge BCont(p, x) \wedge BCont(q, x) \wedge BCont(r, x) \wedge \forall s(sf\_line(s) \wedge BCont(s, x) \rightarrow s = p \vee s = q \vee s = r)]$ (***sf_triangle is a sf_polygon with three linear edges***)

(SFC-M14) $sf\_tin(x) \leftrightarrow sf\_polyhedral\_surface \wedge \forall y[Min(y) \wedge PP(y, x) \rightarrow sf\_triangle(y)]$

(***sf_tin is an aggregation of sf_triangles***)

4.2.2 Axiomatization of Simple Feature's Simple Feature Collections

Sf_multi_point is equivalent to CODI's *PointRegion* (SFC-M15) and is an aggregation of *sf_point*s. *Sf_multi_curve* is equivalent to CODIB's *Multipart_Curve* whose minimal parts are *sf_curve*s (SFC-M16), and it generalizes *sf_multi_line_string* that has *sf_line_string*s as its minimal parts (SFC-M18). A *sf_multi_surface* is equivalent to CODIB's *Multipart_ArealRegion* and is an aggregation of *sf_surface*s (SFC-M17). Its specialization *sf_multi_polygon* aggregates *sf_polygon*s (SFC-M19). A *sf_multi_curve* or *sf_multi_surface* is simple if and only if all of its elements are simple, but it can also be branched where intersections occur between more than two elements along a common boundary.

(SFC-M15) $sf\_multi\_point(x) \leftrightarrow PointRegion(x) \wedge \forall y[PP(y, x) \rightarrow sf\_point(y)]$

(***sf_multipoint is equivalent to CODI's PointRegion***)

(SFC-M16) $sf\_multi\_curve(x) \leftrightarrow Multipart\_Curve(x) \wedge \forall y[P(y,x) \wedge Min(y) \rightarrow sf\_curve(y)]$

(***sf_multicurve is equivalent to CODIB's MultipartCurve whose minimal parts are sf_curves***)

(SFC-M17) $sf\_multi\_surface(x) \leftrightarrow Multipart\_ArealRegion(x) \wedge \forall y[P(y,x) \wedge Min(y) \rightarrow sf\_surface(y)]$ (***sf_multisurface is equivalent to CODIB Multipart_ArealRegion whose minimal parts are sf_surfaces***)

(SFC-M18) $sf\_multi\_line\_string(x) \leftrightarrow sf\_multi\_curve(x) \wedge \forall y[P(y,x) \wedge Min(y) \rightarrow sf\_line\_string(y)]$ (***sf_multilinestring is a sf_multicurve with minimal parts that are sf_linestrings***)

(SFC-M19) $sf\_multi\_polygon(x) \leftrightarrow sf\_multi\_surface(x) \wedge \forall y[P(y,x) \wedge Min(y) \rightarrow sf\_polygon(y)]$

(***sf_multipolygon is a sf_multisurface with minimal parts that are sf_polygons***)

The axioms of SFC-Core together with the mappings SFC-M1 to SFC-M18 form the ontology SFC-FOL[7] . The theorems SFC-T1 to SFC-T2 can be proved from SFC-FOL.

4.2.3 Axiomatization of Simple Feature's Qualitative Spatial Relations

So far we have focused on elaborating the semantics of SFA's feature types using CODIB. But SFA's mereotopological relation can, likewise, be expressed using CODIB's relations as summarized in Table 4.1, similar to the mapping between the DE-I9 relations and CODI [131]. All SFA relations, except for *sf_disjoint*, are specializations of contact (C). *Sf_disjoint* is the negation of contact (SFR-M1), which places no dimensional restriction on the involved entities. The relation *sf_touches* relates two connected features who share parts of their boundaries (i.e. $\partial x \cap \partial y \neq \emptyset$) but no parts of their interiors ($x^\circ \cap y^\circ = \emptyset$). This specializes CODIB's superficial contact relation SC that holds for objects that are in contact but do not share a part of either object. But SC is not sufficient as it allows the lower-dimensional entity

[7]Available from `https://colore.oor.net/simple_features`.

to share part of its interior with the higher-dimensional entity (e.g. a curve segment tangential to a region). Instead, *sf_touches* needs to express that any shared entities are boundary contained in both of the participating entities (SFR-M2). Then, *SC* becomes provable from it (SFR-T1). From the definition of *SC* it can further be inferred that *sf_touches* applies to entities of any dimension except between two points (SFR-T2).

Sf_crosses is a specialization of one of two of CODIB's relation: (1) incidence *Inc* for two entities of different dimension, where a part of the lower-dimensional entity is contained in the higher-dimensional one (e.g. a curve being incident with a polygon by a segment of the curve being contained in the polygon), or (2) superficial contact *SC* for two entities of equal dimension that share only a lower-dimensional entity (e.g. two curves intersecting in a point) (SFR-M3).

Sf_overlaps is a stronger contact relation that only applies to two equidimensional entities and is equivalent to CODIB's partial overlap *PO* when neither entities is a part of the other (SFR-M4). Full containment of an entity inside another entity of the same spatial dimension is represented in CODI by its primitive containment relation, which maps to *sf_contains* (SFA-M5) and to *sf_within* for its inverse (SFR-M6). The special case of spatial equality is captured by *sf_equals* (SFR-M7). *sf_intersects* is the negation of *sf_disjoint* (SFR-M8), which means it generalizes *sf_touches*, *sf_crosses*, *sf_overlaps*, *sf_contains*, *sf_within*, and, indirectly, *sf_equals* (SFR-T6) and is logically equivalent to CODIB's contact relation (SFR-T7). *sf_relate* describes any of SFA's mereotopological relations (SFR-M9), which maps to any pair of spatial entities in CODIB no matter how they are spatially related (SFR-T8).

The axioms of SFC-FOL together with the mappings SFR-M1 to SFR-M9 form the ontology SFR-Core[8] . The theorems SFR-T1 to SFR-T8 can be proved from SFR-FOL.

[8]Available from `https://colore.oor.net/simple_features`.

SFA	9IM	Definition in terms of CODIB relations and additional theorems
disjoint	disjoint	**(SFR-M1)** $sf\_disjoint(x, y) \leftrightarrow S(x) \wedge S(y) \wedge \neg C(x, y)$
touches	meet	**(SFR-M2)** $sf\_touches(x, y) \leftrightarrow S(x) \wedge S(y) \wedge \forall z[Cont(z, x) \wedge Cont(z, y) \rightarrow BCont(z, x) \wedge BCont(z, y)]$ **(SFR-T1)** $sf\_touches(x, y) \rightarrow SC(x, y)$ **(SFR-T2)** $sf\_touches(x, y) \rightarrow sf\_point(x) \wedge \neg sf\_point(y)$
crosses	-	**(SFR-M3)** $sf\_crosses(x, y) \leftrightarrow S(x) \wedge S(y) \wedge [[Inc(x, y) \wedge \neg Cont(x, y) \wedge \neg Cont(y, x)] \vee \forall z[Cont(z, x) \wedge Cont(z, y) \rightarrow Curve(x) \wedge Curve(y) \wedge (z <_{\text{dim}} x \wedge z <_{\text{dim}} y \wedge \neg BCont(z, x) \wedge \neg BCont(z, y)]]$ **(SFR-T3)** $x <_{\text{dim}} y \wedge sf\_crosses(x, y) \rightarrow Inc(x, y) \wedge \neg Cont(x, y)$ **(SFR-T4)** $x =_{\text{dim}} y \wedge sf\_crosses(x, y) \rightarrow SC(x, y)$ **(SFR-T5)** $sf\_crosses(x, y) \wedge sf\_curve(x) \wedge sf\_curve(y) \rightarrow SC(x, y)$
overlaps	overlap	**(SFR-M4)** $sf\_overlaps(x, y) \leftrightarrow S(x) \wedge S(y) \wedge PO(x, y) \wedge \neg P(x, y) \wedge \neg P(y, x)$
contains	contains/ covers	**(SFR-M5)** $sf\_contains(x, y) \leftrightarrow S(x) \wedge S(y) \wedge Cont(x, y)$
within	inside/ coveredBy	**(SFR-M6)** $sf\_within(x, y) \leftrightarrow sf\_contains(y, x)$
equals	equal	**(SFR-M7)** $sf\_equals(x, y) \leftrightarrow sf\_contains(x, y) \wedge sf\_within(x, y)$
intersects	¬ disjoint	**(SFR-M8)** $sf\_intersects(x, y) \leftrightarrow \neg sf\_disjoint(x, y)$ **(SFR-T6)** $sf\_intersects(x, y) \leftrightarrow sf\_touches(x, y) \vee sf\_crosses(x, y) \vee sf\_overlaps(x, y) \vee sf\_contains(x, y) \vee sf\_within(x, y)$ **(SFR-T7)** $sf\_intersects(x, y) \leftrightarrow S(x) \wedge S(y) \wedge C(x, y)$
relate (any)	-	**(SFR-M9)** $sf\_relate(x, y) \rightarrow sf\_intersects(x, y) \vee sf\_disjoint(x, y))$ **(SFR-T8)** $sf\_intersects(x, y) \leftrightarrow S(x) \wedge S(y)$

Table 4.1: SFA's mereotopological relations, their equivalent *Egenhofer* relations, and the developed mappings to CODIB's relations. The relations in the bottom part are all defined in terms of the top five relations.

4.3 Logical Verification

Our primary tool for evaluating the developed first-order ontology SF-FOL are different variants of consistency checking summarized in Table 4.2. In its simplest form, consistency checking verifies that an ontology is free of internal contradiction. This typically involves constructing some small finite model using a finite model finder. A known problem with this approach is that it aims to construct the smallest models, which are often trivial in the sense that the extension of many classes and relations therein are empty or universal. For example, one trivial model for CODIB consists of a set of isolated points, but without any curves or areal regions. Moreover, most of the CODIB relations, such as *BCont*, *SC*, or *Inc*, may not be used at all in a trivial model whereas other relations, such as *Cont* or P, may relate objects only to themselves. Such a model does not prove that all classes may indeed be instantiated (i.e. some curve, areal region, or more specialized defined subclasses such as a branched curve) and all relation may apply to pairs of distinct entities. One can force the creation of non-trivial models by adding existential axioms of the form $\exists x P(x)$ and $\exists x, y[R(x, y) \wedge x \neq y]$ to the ontology. This approach has been implemented in the Macleod suite of tools[9] and previously been utilized to prove CODI's and CODIB's nontrivial consistency with the help of the finite model finder Paradox [56]. Here, the same approach is used to prove SF-FOL's nontrivial consistency.

An additional way to verify an ontology is to prove its consistency with some sample datasets. Rather than constructing an arbitrary model that satisfies certain constraints, this external verification ensures that the ontology is actually consistent with the kind of model encountered in the domain. This has not been done previously for CODI or CODIB as real-world purely qualitative information is hard to come by. However, by mapping SFA concepts to CODIB as a qualitative generalization thereof, we can now exploit the abundance of geometric data already stored in GIS or geospatial databases.

In this work SF-FOL is verified internally, nontrivially and externally with Paradox. Proving nontrivial consistency of SF-FOL ensures that instantiation of all the axiomatically defined or restricted Simple Feature types and SFA's mereotopological relations is possible and the new mappings and axioms do not contain any contradictions. In addition, we employed small subsets of data, consisting of samples of 20 to 40 geometric features, to externally

[9] `https://github.com/thahmann/macleod`

Type	Task	Description
Internal verification	Consistency checking	Ascertains the ontology is free of internal contradictions
	Non-trivial consistency checking	Ascertains that a model exists that instantiates each class and each relation positively and negatively by pairs of distinct objects
External verification	Consistency checking with data	Ascertains that the ontology is consistent with a set of assertions describing a dataset

Table 4.2: Overview of the employed consistency checking methods for the verification of SF-FOL.

verify SF-FOL. The data is extracted from publicly hosted shapefiles[10] that includes polygon representations of counties and subdivisions, polyline representations of major roads, and point representations of schools and other civic buildings within the state of Maine. Only the type of geometry and the SFA relations to other, nearby geometries are stored as assertions. The extracted assertions (i.e. the ABox) were added to SF-FOL (i.e. the TBox) and handed to the model finder to construct a model (external verification results are provided in Chapter 7). As an additional step, we encoded sample queries, such as '*What are the areal regions within Penobscot county that intersect I-95?*', which can be expressed logically in CODIB as $ArealRegion(s) \wedge sf\_within(s,'PenobscotCounty') \wedge sf\_intersects(x,'I95')$. This allows retrieving possible instantiations of x, which were manually inspected to identify any unintended models, such as schools being returned as possible solutions, that helped refine the axiomatization.

Generally, the utilized ontology verification techniques are somewhat similar to software testing techniques: they can help identify problematic models of an ontology that require changing or adding axioms but do not prove that the ontology is fully correct. This would require a full representation theorem describing the structure of *all the models* of SF-FOL,

[10] `https://www.maine.gov/megis/catalog/`

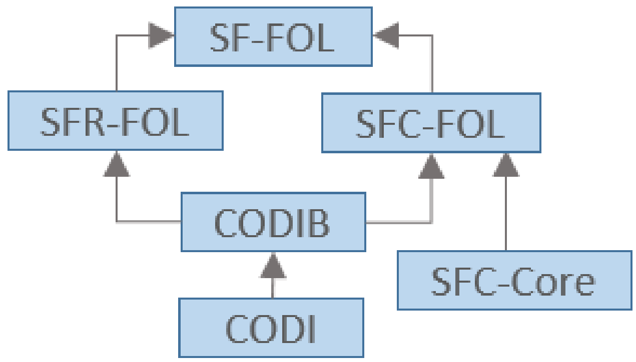


Figure 4.3: The relationships between the developed and reused axiomatic theories.

which is beyond the scope of this work. The completeness of SF-FOL is not verified as this would require alternative characterization of all models.

4.4 Discussion

A core component of many geospatial data models and standards used to store and analyze conventional GIS data are taxonomic classifications of geometric feature types and basic mereotopological relations to support qualitative querying of the geometric data. However, the semantics of the mereotopological relations are not explicitly formalized and thus not accessible for further automated reasoning. Because of this limitation, purely qualitative spatial information, i.e. spatial information that relates objects for which no geometric information is available in the data store, cannot be easily reasoned over in conjunction with existing geometric data. To address this challenge, this chapter presents a semantically augmented formalization, SF-FOL, of the basic geometric feature types (axiomatized in SFC-FOL) and qualitative spatial relations (axiomatized in SFR-FOL) of the Simple Features Access (SFA) standard. This augmented formalization is provided as an extension of the CODIB theory, a qualitative axiomatization of mereotopological space in first-order logic. The relationships between the developed theories is illustrated in Figure 4.3.

It is shown that all of SFA's geometric features specialize the more general, only dimensionally-constrained, classes of spatial entities from CODIB and its subtheory CODI.

The distinctions between "straight line segments" and "curve segments" and, analogously, between "fully bounded regions" and "polygons" are the only ones that are not fully definable in CODIB because they are inherently geometric[11] . But because these distinctions are irrelevant to mereotopological relations, all of CODIB's spatial relations can be evaluated over geometric features in SF-FOL. Likewise, all of SFA's mereotopological relations are fully defined in the SFR-FOL module of SF-FOL and thus can be employed for querying over both geometric and qualitative data.

[11]One cannot distinguish a straight line from a curve without a metric in the space that defines the shortest segment between two points, see the discussion of such issues in [45, 132]

CHAPTER 5

THE ROLE OF AN ONTOLOGY'S SIGNATURE IN SAT-BASED MODEL FINDING

More and more FOL ontologies are becoming available, ranging from upper ontologies such as DOLCE or GFO to ontologies for space, processes, and the geosciences including the axiomatization of qualitative and geometric space presented in Chapter 4. Such ontologies are developed with the intention to enable automated reasoning tools to efficiently infer reliable information with data for decision making, or in the absence of data to prove theorems or lemmas within the domain. Ontology verification through internal consistency checking, and ontology validation through external consistency checking with real-world data are key to making accurate inferences. For consistency checking, traditional model finders (e.g. Paradox [56] or Vampire [172]) translate the FOL problem into an equivalent propositional satisfiability (SAT) problem in Clausal Normal Form (CNF) and then use a SAT solver to determine satisfiability through the generation of a model. To search for models, these model finders instantiate the CNF formula corresponding to the ontology with (an increasing number of) individuals to produce a series of SAT problems, whose size (as measured in the number of propositional variables and clauses) grows exponentially with the number of individuals in the model and the size of the ontology's terminology (number and arity of predicates). While SAT solvers have been found to capably handle large SAT problems, they often experience scalability issues when trying to construct models for FOL ontologies. Available model finders, such as Paradox [56] or Mace4 [200], have been mostly tested on relatively small axiomatizations with few nonlogical symbols (i.e. predicate and function symbols), as commonly found in mathematical conjectures but not representative of ontologies. But even for FOL ontologies with relatively modestly sized terminologies, the results reported in the literature [239, 23, 46] are rather discouraging, with models rarely exceeding 20 individuals, because the program either runs out of memory or never terminates.

Extremely important to SAT solver efficiency are mechanisms that reduce the size of the input CNF formula in order to reduce the time and memory used. Traditional methods of complexity computation of SAT algorithms [215, 101] have relied on measuring the required amount of resources as a function of the input problem's size, specifically the number of clauses and variables [53, 247, 215]. But almost all these studies focus on original SAT problems and not SAT translations of FOL problems. Secondly, as pointed out in [236] there is often a vast discrepancy between theoretical performance and practical performance of SAT solvers, due to the fact that complexity is determined solely based on the general structure of the problem [93], but ignore other structural properties, which may arise from the nature of the domain but also the language – for example concepts in certain FOL axiomatizations may be structured like a list, whereas in some they may assume a tree structure with a root and dependent concepts. SAT solvers are usually considered to be black boxes – when a first-order logic problem is translated to a CNF formula for SAT-based model finding, most of its axiomatization-based structure is already lost, for example dependency between predicates. Therefore, intuitions about the problem domain are no longer accessible to help solve the resulting SAT problem. Research studying the correlation between the signature of an ontology and the hardness of SAT solving is scarce. In particular, SAT heuristics are usually not concerned with the arity of 'FOL-literals', which is shown in this chapter to contribute to an exponential search space in FOL-CNF formulas[1] . There is also no existing work that studies how certain structural dependencies within an FOL ontology can be exploited to simplify an FOL ontology with data leading to a reduction in the size of its SAT translation to improve the scalability of model finding. To bridge these gaps, this chapter develops a formal treatise of ontologies with data, and their CNF translations, and techniques for reducing their size.

Towards the overarching objective of this dissertation, which is to enable integrated spatial reasoning with datasets, and specifically towards objective 2 (O2 in Section 1.2.2) of this

[1]From here on, an FOL-CNF 'formula' refers to the CNF translation of an entire FOL ontology,

dissertation, in this chapter we study the hardness of model finding of data-incorporated FOL ontologies and make the following contributions:

1. Develop a formal account of FOL ontologies with data and define various '*size*' measures on its corresponding CNF and SAT translations.

2. These formalized terms are used to illustrate the growth in size of the FOL-CNF and SAT representations with the ontology's signature, which is identified as a key contributing factor to the dramatic growth of the resulting SAT problems.

3. Develop and define a simplifying heuristic called *Optional Definition Elimination* - ODE, that eliminates select predicates from an FOL ontology before their translation to SAT.

ODE is a variant of definition inlining implemented in VAMPIRE's clausfier [229]. We formalize this formula simplification ODE_D (D is the set of optional definitions for elimination) in this chapter and implement it as an FOL preprocessing technique in Chapters 6 and 7, where we test the following hypothesis "*removal of additional defined terms from an FOL ontology can significantly improve SAT model finding performance in practice.*"

5.1 SAT-Based Model Finding for FOL Ontologies

The Mace-style finite-model building approach [200, 56, 269] used in popular automated theorem provers (ATPs) such as Paradox [56], Vampire [172] and iProver [165] works by converting a first-order logic ontology into a set of propositional logic sentences and handing them off to a SAT-solver. Thus, FOL model finding is a two-staged process as shown in Figure 5.1.

5.1.1 Size of the Clausified FOL Ontology

Through applying Skolem's algorithm from [29] an FOL formula can be translated to a quantifier-free formula in CNF. This translation converts the formula to existential-quantifier-free (universally quantified and so the outside quantifiers can be removed), function-free FOL formula through (1) the elimination of conditionals and biconditionals, (2) pushing negations inwards, (3) standardizing and renaming variables, (4) skolemization, (5) eliminating quantifiers, and (6) distributing disjunctions using De Morgans laws (described in detail in Section 2.2.2). This resulting FOL-CNF formula may introduce additional constants and functions and therefore is not equivalent but equi-satisfiable with the original FOL formula. The FOL-CNF formula corresponding to an ontology $\mathcal{O}$ is defined in Def. 10 as $\mathcal{O}_{\text{FOL-CNF}}$.

Definition 10. *Let $\mathcal{O}$ be an FOL ontology. Then we call its FOL-CNF formula obtained through the 7 clausification steps from* [29] *$\mathcal{O}_{FOL\text{-}CNF}$. This FOL-CNF representation is in clausal normal form (CNF) whose variables are all universally quantified.*

The size of the signature of $\mathcal{O}_{\text{FOL-CNF}}$ is defined in terms of the number of predicates of each arity as follows:

Definition 11. *Let $\mathcal{O}_{FOL\text{-}CNF}$ be an ontology's FOL-CNF representation. Then*

- *$sf_{a=n}$ denotes the set of n-ary Skolem functions introduced by skolemization*[2] *. If treated as predicates, the set $sf_{a=n}(\mathcal{O}_{FOL\text{-}CNF})$ adds that many $(n+1)$-ary predicates to $\mathcal{O}_{FOL\text{-}CNF}$.*
- *$\Omega_{a=n}(\mathcal{O}_{FOL\text{-}CNF}) = \{\Omega \in \lambda(\mathcal{O}) \mid a(\Omega) = n)\} \cup sf_{a=n-1}(\mathcal{O}_{FOL\text{-}CNF})$ defines the set of predicates of arity n, which includes the n-ary predicates from $\mathcal{O}$ as well as any newly introduced $(n-1)$-ary Skolem functions.*

The size of $\mathcal{O}_{\text{FOL-CNF}}$ itself is defined in terms of its number of clauses aand other measures defined as follows:

Definition 12. *Let $\mathcal{O}_{FOL\text{-}CNF}$ be an ontology's FOL-CNF representation treated as set of clauses. Then,*

- *for any single clause $C \in \mathcal{O}_{FOL\text{-}CNF}$, the <u>clause-width</u> $w(C)$ is the number of FOL literals therein.*

[2]In our work, n is at most 2, However, depending on the number of nested quantifiers of the original ontology, Skolem functions of higher arity may be introduced.

- *the formula-width of $\mathcal{O}_{FOL\text{-}CNF}$ is the maximal clause-width of all clauses in $\mathcal{O}_{FOL\text{-}CNF}$, defined as $W(\mathcal{O}) = \max\{w(C)|C \in \mathcal{O}_{FOL\text{-}CNF}\}$.*

- *for any single clause $C \in \mathcal{O}_{FOL\text{-}CNF}$, the variable-density is the distinct number of FOL variables therein.*

- *the maximal variable-density of all clauses in $\mathcal{O}_{FOL\text{-}CNF}$ is given by v^*.*

The translation of a first-order logic formula to an FOL-CNF formula is demonstrated by the following example.

Example 1. *Consider a small ontology $\mathcal{O}_{RCC\text{-}s}$ with three sentences (1 axiom and 2 definitions), this is a subset of the FOL axiomatization of the RCC and the signature $\lambda(\mathcal{O}_{RCC\text{-}s}) = \{C, P, PP\}$ denoting contact $C(x, y)$, parthood $P(x, y)$, and proper parthood $PP(x, y)$.*

$$(\sigma_C)\ \ C(x, y) \rightarrow C(y, x)$$

$$(\sigma_P)\ \ P(x, y) \leftrightarrow \forall z[C(z, x) \rightarrow C(z, y)]$$

$$(\sigma_{PP})\ \ PP(x, y) \leftrightarrow P(x, y) \wedge \neg P(y, x)$$

Following clausification, the FOL-CNF formula for $\mathcal{O}_{RCC\text{-}s}$ has seven clauses ($C = 7$) where 2 clauses have width $w = 3$ and 5 clauses have width $w = 2$ each (see Table 5.1), where the width denotes the number of FOL literal in a clause (as defined in Def. 12). Skolemization introduces one additional binary function - f, resulting in a total of 3 binary and 1 ternary predicate ($|\Omega_{a=3}| = 1$, $|\Omega_{a=2}| = 3$) in the FOL-CNF representation.

Note: For conceptual simplicity each n-ary function symbol is treated as an $n + 1$-ary predicate symbol. Therefore following skolemization each unique Skolem constant is treated as a unary predicate, each unique unary function is treated as a binary predicate and so on.

5.1.2 Size of the Propositionalized FOL-CNF Ontology

The second step in propositionalization involves instantiating all variables within the FOL-CNF clauses over all combinations of individuals from a fixed domain. This first requires

Clause 1	$\neg c(x,y) \vee c(y,x)$.
Clause 2	$\neg p(x,y) \vee \neg c(z,x) \vee c(z,y)$.
Clause 3	$p(x,y) \vee c(f(x,y),x)$.
Clause 4	$p(x,y) \vee \neg c(f(x,y),y)$.
Clause 5	$\neg pp(x,y) \vee p(x,y)$.
Clause 6	$\neg pp(x,y) \vee \neg p(y,x)$.
Clause 7	$pp(x,y) \vee \neg p(x,y) \vee p(y,x)$.

Table 5.1: FOL-CNF clauses for the three sentences in $\mathcal{O}_{RCC\text{-}s}$. Clauses are separated by conjunctions.

fixing the domain size (i.e. the number of distinct individuals) [279]. If the domain size is not known in advance, the model finder starts with domain size 1 and incrementally increases it each time the search space is exhausted. If, for example, the smallest model has 8 individuals, then the model-finder will run 7 SAT instances that are proved to be unsatisfiable and an 8th one that is satisfiable. The propositional representation of an ontology $\mathcal{O}$ instantiated for a domain d is defined in Def. 13.

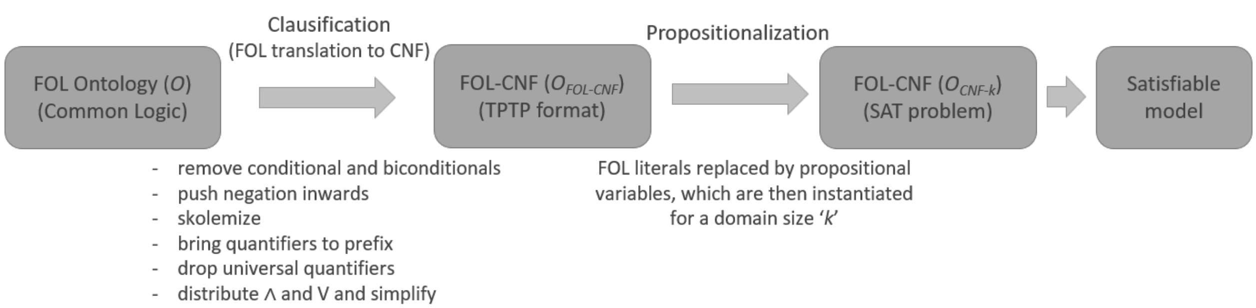


Figure 5.1: Steps involved in the translation of a first-order logic formula to a propositional formula to generate a finite model.

Definition 13. *Let $\mathcal{O}$ be an ontology and $\mathcal{O}_{FOL\text{-}CNF}$ is the FOL-CNF representation, then the propositional instantiation of the ontology with a domain of d individuals is called* $\underline{\mathcal{O}_{CNF\text{-}d}}$.

Every n-ary predicate symbol from the signature of the original ontology will be instantiated into d^n propositional variables. For example, the sentence $Inc(x,y) \vee \neg Lt(z,x) \vee \neg Cont(z,x) \vee$

$\neg P(z, y)$ contains four predicates, and each binary literal, e.g. *Inc(x,y)*, leads to d^2 propositional variables. This is formally captured by the following lemma:

Lemma 1. *Let $\mathcal{O}_{FOL\text{-}CNF}$, be the CNF form of an FOL ontology with maximum arity a^*. Now, the number of propositional variables in its propositional instantiation $\mathcal{O}_{CNF\text{-}d}$ over a domain with d individuals is*

$$P_v = \sum_{i=1}^{a^*} \left(d^i \cdot |\Omega_{a=i}| \right)$$

The number of all propositional clauses (as defined in Section 2.2.1) in $\mathcal{O}_{\text{CNF-d}}$ is denoted by P_c, also referred to as formula-length. Likewise, for a domain size d, each FOL-CNF clause leads to an exponentially growing number d^v of propositional clauses, where v is the number of (implicitly universally quantified) variables in each FOL-CNF clause, because every variable can be independently instantiated with any of the d individuals.

Lemma 2. *Let $\mathcal{O}_{FOL\text{-}CNF}$ be an FOL-CNF ontology where C_v denotes the subset of clauses with v distinct FOL variables per clause, and v^* is the maximal number of variables in any clause in $\mathcal{O}_{FOL\text{-}CNF}$. Then for a domain size d, $\mathcal{O}_{CNF\text{-}d}$ the number of propositional clauses is given by:*

$$P_c = \sum_{i=0}^{v^*} \left(d^i \cdot |C_{v=i}| \right)$$

Thus, the 'size' of the propositional instantiation $\mathcal{O}_{\text{CNF-d}}$ can be jointly described using P_c and P_v: their ratio $r = \frac{P_c}{P_v}$ describes its clause density.

Note: Throughout the rest of this dissertation we adopt this naive approach to calculate P_v and P_c, which are therefore worst-case measures. However, preprocessing techniques built into modern ATPs such as non-ground splitting and symmetry reduction techniques implemented in Paradox, and formula renaming are meant to control the exponential blowup of search space. Nevertheless, our findings will show that these measures are closely correlated to the experimental runtimes of model finders that are presented in Chapter 6.

5.2 SAT-Based Model Finding for FOL Ontologies with Data

For simply proving the consistency of an FOL ontology, no data (*ground facts*) are needed. However, to prove that an ontology is consistent with a given dataset, we need to take the size of a dataset into account when estimating the size of the resulting SAT problem. To investigate how the size of $\mathcal{O}_{\text{CNF-d}}$ changes with different amounts of data in the ontology, we adapt the notions of Terminological Box (TBox), Relations Box (RBox), and Assertion Box (ABox) from Description Logic (DL) ontologies [76, 150]. The TBox captures terminological axioms which constrain the interpretations of concepts (i.e. unary predicates), while the RBox constrains the interpretation of roles (i.e. binary predicates). We will not distinguish between them, but draw the distinction between the TBox (for all terminological axioms) and the ABox, the latter of which captures assertions about individuals, i.e. ground statements about an individual being an instance of a particular concept or being related to another individual via a particular relation.

5.2.1 Assertion Box and Terminological Box

An FOL ontology can mix structural knowledge and assertions about individuals, even in a single sentence. Because the conversion to FOL-CNF tends to separate those at least to some degree, we define an ontology's ABox in terms of the ground formulas in its FOL-CNF version.

Definition 14. *Let $\mathcal{O}$ be an FOL ontology with signature $\lambda(\mathcal{O})$ and let $\mathcal{O}_{FOL\text{-}CNF}$ be its corresponding set of FOL-CNF clauses. Then the* <u>assertion box</u> ABox($\mathcal{O}$) *is the subset of $\mathcal{O}$'s sentences that only yield ground clauses in $\mathcal{O}_{FOL\text{-}CNF}$ that only use symbols from $\lambda(\mathcal{O})$*[3] .

While an ABox may contain disjunctive knowledge – reflected in ground clauses with multiple literals – many clauses are so-called *unit clauses* consisting of only a single literal, which intuitively are *facts*. In the experiments conducted in Chapter 6, we limit the ABox to

[3]Clauses that are ground but use newly introduced Skolem constants or functions are not considered part of the ABox as the Skolem symbols arise from existential quantifiers.

such unit clauses. For simplicity, we further require that the ABox itself, and not just its clausal conversion, is represented as a set of ground clauses. In other words, *the ABox is the dataset* we want to verify an ontology against.

Definition 15. *An* ABox($\mathcal{O}$) *is called* <u>*factual*</u> *iff it contains only unit clauses.*

The spatial ontologies CODI, RCC and INCH contain, like many ontologies, only unary and binary predicates. If the ABox for such an ontology is factual, it consists of three types of assertions:

- **Class Assertions** express membership of an individual in a certain class, e.g. *ArealRegion*('penobscotCounty').

- **Relational Assertions** ascertain two or more individuals to be in a certain relation, e.g. *Inc*('i95', 'penobscotCounty').

- **Distinctness Assertions** ensure that distinct constants denote distinct individuals, e.g. ("i95" $\neq$ "penobscotCounty").

An FOL ontology's TBox captures its structural, i.e. non-factual knowledge. We define it indirectly via the sentences that are not contained in the ABox.

Definition 16. *Let $\mathcal{O}$ be an FOL ontology and* ABox($\mathcal{O}$) *its ABox. Then its* <u>*terminology box*</u> *is defined as* $\mathrm{TBox}(\mathcal{O}) = \mathcal{O} \setminus \mathrm{ABox}(\mathcal{O})$.

For an ontology with a factual ABox, the TBox will not contain any ground clauses except possibly ones involving Skolem symbols.

5.2.2 The Size of SAT Problems for an FOL Ontology with an ABox

In the following example we demonstrate calculating the number of propositional variables and propositional clauses for a $\mathcal{O}_{\text{CNF-d}}$ formula.

Example 2. *Consider the ontology $\mathcal{O}_{RCC\text{-}s}$ with signature $\lambda(\mathcal{O}) = \{C, P, PP\}$ from Example 1. The FOL-CNF version of $\mathcal{O}_{RCC\text{-}s}$ contains 7 clauses with 4 nonlogical symbols, which in addition to the 3 predicates from $\mathcal{O}_{RCC\text{-}s}$ includes one binary Skolem function which is logically representable as a ternary predicate. Propositionalizing the ontology for domain size $d = 20$ yields*

$$P_v = |\Omega_{a=2}| \cdot d^2 + |\Omega_{a=3}| \cdot d^3 = 3 \cdot 20^2 + 1 \cdot 20^3 = 9,200 \textit{ propositional variables.}$$

Out of the 7 clauses, one clause has 3 FOL variables (clause 7 in Table 5.1) while the other six all have 2 FOL variables[4] *.*

Thus the number of propositional variables in the SAT representation is largely dependent upon the number and arity of predicates: each predicate of arity a results in d^a propositional variables for domain size d. This number determines the search space of the propositional SAT problem, which consists (without using any heuristics) of 2^{P_v} possible interpretations. For example, a simple ontology with b binary and u unary predicates (and no other predicates) then yields $(2^b)^{d^2} \cdot (2^u)^d$ interpretations, which is exponential in both the number of binary predicates (and more generally the number of predicates of highest arity) and the domain size d. While modern SAT solvers employ effective strategies to drastically prune the search space and are thus able to deal with thousands of variables and tens of thousands of clauses [116], the growth in P_v and P_c quickly exceeds a million even for ontologies having a modest-sized signature where no predicate has an arity greater than 2 and only a handful of binary predicates included. But this also suggests that improvements can be realized by reducing the total number of predicates, especially those of highest arity. Definition elimination, as formalized in Section 5.3, can achieve this for ontologies with a large number of defined predicates, which can be easily dispensed off before model finding and can be added back in afterwards. But we first look more closely at how the ABox impacts the size of the resulting SAT problem.

[4]Note that FOL variables in different clauses are considered as different variables.

Example 3. *Consider a minimal* ABox($\mathcal{O}$) *with exactly one relational assertion, namely* $PP('m','n')$*. This adds exactly one ground clause (with no FOL variable) to the FOL-CNF formula in Example 2.*

Then for domain size 20 *the propositional version* $\mathcal{O}_{CNF\text{-}20}$ *contains*

$$P_c = |C_{(v=3)}| * d^3 + |C_{(v=2)}| * d^2 + |C_{(v=1)}| * d^1 + |C_{(v=0)}| * d^0$$
$$= 1 \cdot 20^3 + 6 \cdot 20^2 + 0 \cdot 20^1 + 1 \cdot 20^0 = 10,401 \textit{ propositional clauses.}$$

Note: A more general expression for calculating P_v and P_c resulting from an ABox with domain size d and specific number of relational assertions is presented in Lemma 3 in Chapter 6.

5.2.3 Significance of an FOL Ontology's Signature Size for its SAT Encoding

The search space of a SAT problem is often presented as a decision tree, and this space is exponential in $(O(|\Omega||D|^{a^*})$ propositional variables, P_v, for $|\Omega|$ predicates of maximum arity a^* and $|D|$ individuals for every FOL ontology. A standard decision tree[5] has 2^{P_v} leaves – Figure 5.2 represents the basic decision tree for the definition of *PO* from CODI. The width of the tree is therefore bound by the number of propositional variables in $\mathcal{O}_{\text{CNF-d}}$, and each propositional variable in the SAT problem is a potential choice point. P_v in $\mathcal{O}_{\text{CNF-d}}$ is large if and only if $\mathcal{O}_{\text{FOL-CNF}}$ contains large number of predicates (mostly with arity ≥ 2). SAT solving by itself is exponential already (that is, the size of the problem grows exponentially with P_v) but with an FOL-SAT problem P_v also grows exponentially with the size of the FOL signature - which significantly worsens the tractability.

The search performance of a SAT solver is also bound by the number and width of the clauses in $\mathcal{O}_{\text{CNF-d}}$. However, merely the number of clauses is a bad proxy for determining tractability of model finding, as the number of satisfying assignments for a problem is unrelated to the number of clauses. In general, a formula consisting of more clauses will

[5] A binary tree having 2^{P_v} leaves, where the nodes are partial assignments and every leaf is a full assignment.

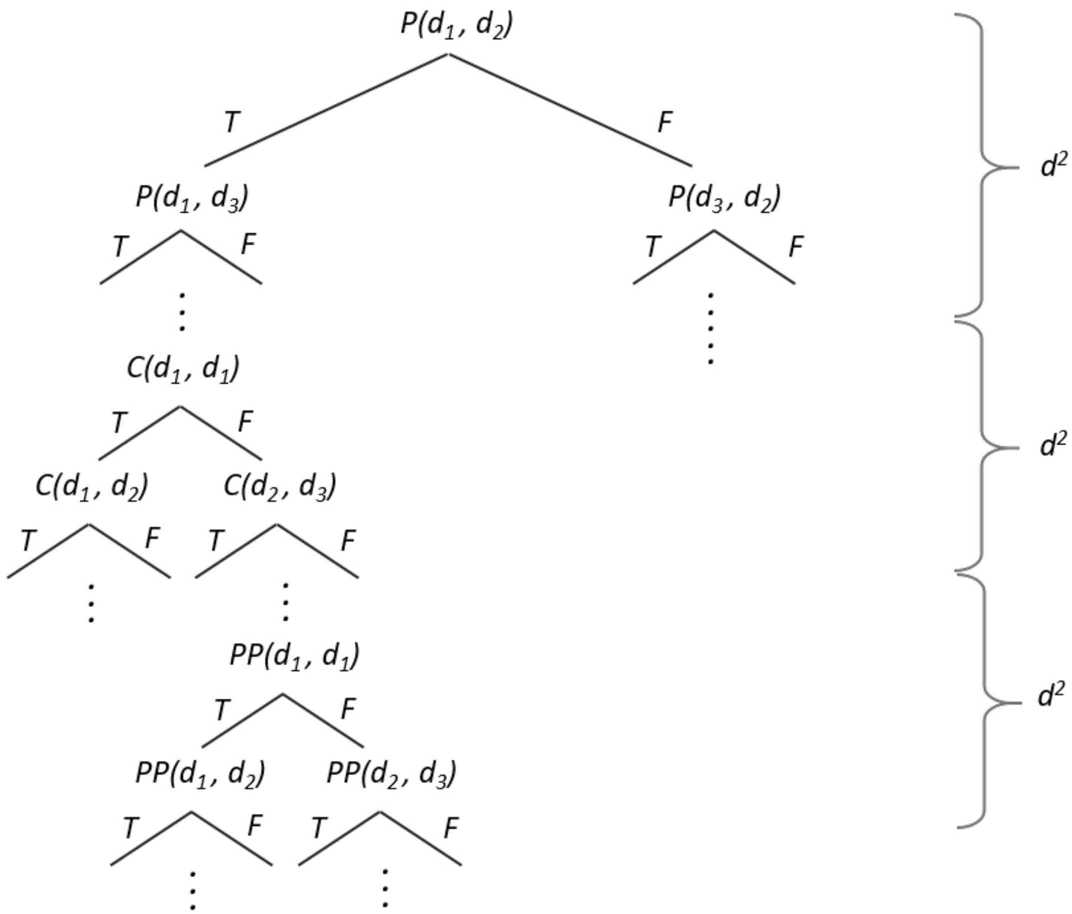


Figure 5.2: Decision tree corresponding to the propositional instantiation of the FOL definition of CODI's PO. To search the space of all truth assignments systematically, both partial and complete, we can instantiate the variables one at a time. The search space is then denoted by: $2^{d^2} \cdot 2^{d^2} \cdot 2^{d^2}$ (since there are three binary predicates in the definition of PO)

.

lead to more conflicts and thus to more frequent backtracking. But at the same time, this backtracking also means potentially more aggressive pruning of the search tree (as for each conflict, a subtree can be pruned). A better measure to predict model finding performance is the width of clauses $w(C)$ in the CNF-formulas, as defined in Def. 12. Specifically, the median width of clauses in $\mathcal{O}_{\text{CNF-d}}$ determines the length of traversal along the node of a tree until a conflict is detected. Each time a propositional variable is assigned, a certain number of clauses are shortened down to unit clauses and eventually empty clauses that represent conflicts. If the average or median length of clauses is higher, it typically will take longer until conflicts are detected.

So, a large ontology signature (predicates and functions) determines the complexity of the problems, and the number and (median) width of clauses determines how fast the saturation algorithm terminates – which is the tractability of the solver. Because of the importance of the size of the decision tree, the tree width and depth of a SAT problem's graph representation, it is very natural to ask about mechanisms to control these parameters for an FOL-CNF formula. Minimizing P_v reduces the search space and, thus, worst-case time for model-finding (since the search space grows exponentially with P_v, a small reduction in the number of predicates – say even by 1 or 2 – can amount to one or several orders of magnitude reduction in P_v). While we anticipate that problems with a higher median width of clauses will also take, at least on average longer than comparably-sized problems with a lower median width of clauses, this is tested in more detail in Chapter 7.

5.3 Definition Elimination for Reducing the Size of the SAT Encodings of FOL Ontologies

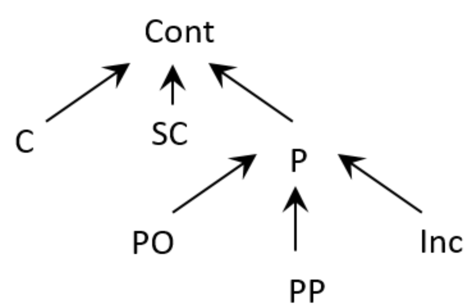


Figure 5.3: Dependency between defined predicates in the CODI ontology.

Now that we have established that the number of predicates with highest arity has an outsized influence on the number of propositional variables in the resulting propositional SAT problem, we illustrate how P_v can be reduced by eliminating defined predicates – and thus reducing the signature overall – before clausifying and propositionalizing an FOL ontology.

We then introduce a formula simplification strategy that exploits the dependency between predicates arising from the manner in which they are formalized to eliminate sets of predicates from an FOL ontology – e.g. Figure 5.3 shows the dependency between defined predicates in the CODI ontology. This simplification strategy is then empirically tested using select spatial ontologies that are at the core of this work in Chapter 6. We identify (1) *optional* definitions as ideal candidate terms for elimination, thereby reducing the size of the propositional formula; (2) axiomatizations that contain many definitions lead to large number of propositional clauses that are generated by converting biconditionals to clausal form, and postulate that by bypassing this we can improve model finding.

The following example shows how reducing the signature of an ontology alters the size of its SAT representation.

Example 4. *We reuse the* TBox *from Example 1, where* $\lambda = \{C, P, PP\}$ *and one binary Skolem function (analogous to a ternary predicate) is introduced by clausification. Its propositional version contains* $68,800$ *and* $531,200$ *propositional variables for domain sizes* 40 *and* 80*, respectively. Now consider adding another binary predicate* O (*overlap*) *to the signature, which is explicitly defined by*

$$(\sigma_O)\ O(x,y) \leftrightarrow \exists z[P(z,x) \wedge P(z,y)]$$

When adding O*, in the FOL-CNF version, the number of binary predicates increases to* 4 *and the ternary ones to* 2 (*via another binary Skolem function resulting from the existential quantifier in the definition of* O). *Then* P_v *increases to* $134,400$ *and* $1,049,600$ *for* $d = 40$ *and* 80.

The number of FOL-CNF clauses also increases from 7 *to* 10*, with one of the new clauses containing* 3 *variables. Then the number of propositional clauses increases from* $73,600$ *and* $550,400$ *for* $d = 40$ *and* 80 *to* $640,000$ *and* $5,120,000$*, respectively. Note that these measures correspond simply to a TBox in the absence of any relational assertions. In the presence of an ABox, the set of propositional clauses will increase much more.*

In this particular example the addition of just one binary predicate almost doubles P_v while P_c increases eight-fold[6] . But the added predicate O is explicitly defined and thus can be removed before model finding without changing the satisfiability, and its interpretation can be reconstructed for any model later on.

We first define optional definitions, then the DBox as a maximal set of optional definitions that can be easily removed from an ontology, and finally *optional definition elimination* (ODE). Detailed theoretical characterization and empirical evaluation of the effect of ODE on the size of propositional SAT problems arising from FOL model finding is studied in Chapter 6.

Definition 17. *A <u>substitution</u> is a mapping $\alpha : V \to T$ from variables (in a term or a formula) to terms.*

- *Term substitution is the result of substituting term t in term s for a term x, denoted by $s[t/x]$ and is defined recursively as follows: $y[t/x]$ = if $y \neq x$ then y else t, when s is a variable y; $c[t/x] = c$, when s is a constant c, called ground substitution; $ft_1...t_n[t/x] = ft_1[t/x]...t_n[t/x]$, when s is a term $ft_1...t_n$.*
- *Formula substitution $F[t/x]$ can be defined similarly for a formula F to replace all free occurrences of t with x in the formula F.*

An explicit definition [32] of a predicate is a special type of TBox sentence

Definition 18. *Let $\mathcal{O}$ be an ontology with signature $\lambda(T)$. Then an <u>explicit definition</u> of an n-ary predicate $\Omega \in \lambda(\mathcal{O})$ in an ontology $\mathcal{O}$ is a sentence $\sigma \in \mathrm{TBox}(O)$ of the form*

$$\forall x_1, \ldots, x_n[\Omega(x_1, \ldots, x_n) \leftrightarrow \alpha(x_1, \ldots, x_n)]$$

wherein α is a formula with x_1 to x_n as only free variables and with $\lambda(T) \setminus \Omega$ as the only nonlogical symbols. Then Ω is said to be <u>explicitly defined</u> in T.

[6]This variation is based on the nature of the axiomatization. Depending on the complexity of newly added formulas, the increase can be moderate or exceptionally large.

Optional definitions are explicit definitions of predicates that are not used in other sentences of the ontology's TBox:

Definition 19. *An explicit definition* $\sigma \in \mathrm{TBox}(\mathcal{O})$ *of a symbol* $\Omega \in \lambda(\mathcal{O})$ *is an* <u>*optional definition*</u> *in* $\mathcal{O}$ *iff* Ω *does not appear in any sentence in* $\mathrm{TBox}(\mathcal{O}) \setminus \sigma$*.*

Now we can recursively define larger *definitions sets*, with the maximal one being referred to as the ontology's DBox:

Definition 20. *A* <u>*definition set*</u> *of an ontology* $\mathcal{O}$ *is defined recursively as:*

B. *The set of all optional definitions in* $\mathrm{TBox}(\mathcal{O})$ *forms a definition set;*

R. *For any definition set* D *of* $\mathcal{O}$ *and for any optional definition* σ *of* Ω *in* D*, the set* D' *defined as follows is a definition set:* $D' = D' \cup \sigma | \sigma \in D$*,*
that is, D' *is constructed recursively by adding* σ *as a new definition to the set.*

Definition 21. *For an ontology* $\mathcal{O}$*,* <u>*DBox($\mathcal{O}$)*</u> *is a definition set such that no optional definition exists in* $\mathrm{TBox}(\mathcal{O}) \setminus \mathrm{DBox}(\mathcal{O})$*.*
$\Omega \in \lambda(T)$ *is* <u>*optionally defined*</u> *in* $\mathcal{O}$ *iff* Ω *does not appear in* $\mathrm{TBox}(\mathcal{O}) \setminus \mathrm{DBox}(\mathcal{O})$*.*

To study how removing optionally defined predicates impacts the size of the SAT representation, we also need to substitute the eliminated predicates in the ABox without changing the ontology's semantics. This is achieved by replacing assertions that use optionally defined predicates by *defined assertions*.

Definition 22. *Let* $\mathcal{O}$ *be an ontology and* D *some definition set of* $\mathcal{O}$*.*
Then <u>*ABox$_D$($\mathcal{O}$)*</u> $= \mathrm{ABox}(\mathcal{O})\Big[\bigcup_{\sigma_i \in D}[\Omega_i(x_1, \ldots, x_n)/\alpha_i(x_1, \ldots, x_n)]\Big]$.
Any sentence $\sigma \in \mathrm{ABox}_D(\mathcal{O})$ *with* $\sigma \notin \mathrm{ABox}(\mathcal{O})$ *is called a* <u>*defined assertion*</u>*.*

In other words, $ABox_D(\mathcal{O})$ is $\mathcal{O}$'s ABox with all occurrences of predicates Ω_i that are optionally defined by some definition in D (which typically would be the entire DBox of $\mathcal{O}$) substituted by their definiens α_i. Note that an ABox with defined assertions may no longer only contain only ground unit clauses. Defined assertions may contain variables introduced during the substitution. For example, a fact $O(\text{‘i95’}, \text{‘295w’})$ would result in the defined assertion $\exists z[P(z, \text{‘i95’}) \wedge P(z, \text{‘295w’})]$ if O is substituted by the definition from Example 4.

By how we remove optional definitions only and substitute their occurrences in the ABox , the satisfiability of the ontology remains unchanged. This follows directly from the well-known relationship between explicit and implicit definability (Beth's definability theorem [32]) and is captured by the following theorem:

Theorem 1. *Let $\mathcal{O}$ be an FOL ontology and D be a definition set of $\mathcal{O}$. Then there is a bijection between the models of $\Big(\mathrm{TBox}(\mathcal{O}) \setminus D\Big) \cup \mathrm{ABox}_D(\mathcal{O})$ and the models of $\mathrm{TBox}(\mathcal{O}) \cup \mathrm{ABox}(\mathcal{O})$, that is, every model of $\Big(\mathrm{TBox}(\mathcal{O}) \setminus D\Big) \cup \mathrm{ABox}_D(\mathcal{O})$ can be uniquely expanded into a model of $\mathrm{TBox}(\mathcal{O}) \cup \mathrm{ABox}(\mathcal{O})$.*

Proof. Note that from the construction of $\mathrm{ABox}_D(\mathcal{O})$ in Def. 22, $D \cup \mathrm{ABox}_D(\mathcal{O}) \equiv \mathrm{ABox}(\mathcal{O})$. Further note that D explicitly defines the set of symbols in $\lambda(\mathcal{O})$ but not used in $\Big(\mathrm{TBox}(\mathcal{O}) \setminus D\Big) \cup \mathrm{ABox}_D(\mathcal{O})$. Then $\Big(\mathrm{TBox}(\mathcal{O}) \setminus D\Big) \cup \mathrm{ABox}_D(\mathcal{O}) \cup D \equiv \mathrm{TBox}(\mathcal{O}) \cup \mathrm{ABox}(\mathcal{O})$.
We can then apply Beth's definability theorem [32], which established a correspondence between explicit definability of a term in FOL and implicit definability of the same terms in a structure. Since here the predicates defined by D are explicitly definable in $\Big(\mathrm{TBox}(T) \setminus D\Big) \cup \mathrm{ABox}_D(T)$, they are implicitly definable in its models, which become models of $\mathrm{TBox}(T) \cup \mathrm{ABox}(T)$ by the logical equivalence of the two theories.

$\square$

The DBox captures the maximal set of optional definitions that can be easily removed without altering the ontology's semantics.

Corollary 1. *Let $D = \mathrm{DBox}(\mathcal{O})$. Then there are bijections between the models of $\mathcal{O} = \mathrm{TBox}(\mathcal{O}) \cup \mathrm{ABox}(\mathcal{O})$ and $\Big(\mathrm{TBox}(\mathcal{O}) \setminus D\Big) \cup \mathrm{ABox}_D(\mathcal{O})$. And therefore, $\mathcal{O} = \mathrm{TBox}(\mathcal{O}) \cup \mathrm{ABox}(\mathcal{O})$ is satisfiable iff $\Big(\mathrm{TBox}(\mathcal{O}) \setminus D\Big) \cup \mathrm{ABox}_D(\mathcal{O})$ is satisfiable.*

The model are not the same because they use different signatures, but there is a mapping between them. This idea forms the basis of our strategy for improving model finding because $\Big(\text{TBox}(T) \setminus \text{DBox}(T)\Big)$ has a smaller signature than $\mathcal{O} \equiv \text{TBox}(\mathcal{O}) \cup \text{ABox}(\mathcal{O})$ but is equi-satisfiable. These formal results (Theorem 1 and its corollary) inform ODE as a technique.

Definition 23. *Let $\mathcal{O}$ be an FOL ontology, with a factual* $\text{ABox}(\mathcal{O})$*, and let D be a definition set of $\mathcal{O}$. D could be the equal to* $\text{DBox}(\mathcal{O})$ *or* $D \in \text{DBox}(\mathcal{O})$*. Then ODE can be applied to obtain an equi-satisfiable ontology $\mathcal{O}'$ in the following way:*

- *For every $\sigma_d \in D$, all sentences $\sigma_a \in \text{ABox}(\mathcal{O})$ that use Ω of σ_d is replaced with its defined assertion and then σ_d is removed from* $\text{TBox}(\mathcal{O})$*.*

The new ABox, $ABox_D(\mathcal{O})$ is called an <u>ODE derived ABox</u>.

Such an ABox may be non-factual and disjunctive. In addition $|ABox_D(\mathcal{O}_{FOL-CNF})| \geq |ABox(\mathcal{O}_{FOL-CNF})|$, i.e. the number of FOL-CNF clauses in the ABox increases after ODE. However this increase will mostly be counteracted by the reduction of FOL-CNF in the TBox from the removal of definitions. This will be examined in more detail in the next chapter.

The following example illustrates the effect of ODE on the size of the resulting SAT problem.

Ontology before ODE	**Ontology after removing PP**
$\text{TBox}(\mathcal{O}) \equiv \text{DBox}(\mathcal{O})$	$\text{TBox}(\mathcal{O}) \equiv (\text{DBox}(\mathcal{O}) \setminus \sigma_{PP})$
σ_P: $P(x,y) \leftrightarrow \forall z[C(z,x) \rightarrow C(z,y)]$ σ_{PP}: $PP(x,y) \leftrightarrow P(x,y) \wedge \neg P(y,x)$	σ_P: $P(x,y) \leftrightarrow \forall z[C(z,x) \rightarrow C(z,y)]$ σ_{PP}: Removed
$\text{ABox}(\mathcal{O})$	$\text{ABox}_D(\mathcal{O})$
β: $PP(\text{`}exit193', \text{`}i95')$	β': $P(\text{`}exit193', \text{`}i95') \wedge \neg P(\text{`}i95', \text{`}exit193')$

Table 5.2: Example of an ontology $\mathcal{O}$ with a TBox and ABox, before and after ODE.

Example 5. *Consider the ontology $\mathcal{O}_{RCC\text{-}s}$ from Example 1.* $\mathrm{DBox}(\mathcal{O}_{RCC\text{-}s})$ *contains two predicates, namely PP and P that are optionally defined, but we use the definition set containing PP to eliminate. Further assume that its ABox still contains β as only assertion (cf. Table 5.2). Now applying ODE only on PP removes σ_{PP} from the TBox and substitutes all occurrences of PP in the ABox with its defined assertion $P(x,y) \wedge \neg P(y,x)$. β will become β'. After ODE, the ontology only has two instead of three binary predicates. For the example domain size* 20, *the propositional problem now contains* $2 * 20^2 + 1 * 20^3 = 8,800$ *propositional variables instead of* $3 * 20^2 + 1 * 20^3 = 9,200$ *as previously. Likewise, the number of propositional clauses is reduced from* $10,400$ *to* $10,000$. *Much larger decreases can be realized by eliminating syntactically more complex definitions, such as the definition of P that contains an existential quantifier. Removing it would eliminate the ternary predicate and lead to a SAT representation with only* $2 * 20^2 = 800$ *propositional variables arising from its TBox*[7] *.*

5.4 Discussion and Conclusion

In this chapter we have presented a formalization of FOL ontologies with data and have identified important parameters for quantifying the size of their FOL-CNF representation: number of predicates and their arity, number of clauses, formula-width, and variable density, thereby addressing objective 2 (O2 in Section 1.2.2) of the dissertation. We then proceeded to demonstrate how the SAT search space, which is bound by the set of propositional variables in the ontology's SAT translation is exponential in the number of predicates of highest arity and domain size. The number of propositional clauses grows polynomially with respect to the number of FOL-CNF clauses but grows exponentially with respect to the highest number of variables in any clause of the FOL-CNF formula and domain size. We have identified these measures as the primary sources of the limitations for model finding with FOL ontologies with larger signatures. We have introduced optional definition elimination technique to eliminate

[7]The number would be larger if the ABox heavily uses the eliminated predicate, as that would reintroduce some variables via Skolemization.

sets of optionally defined predicates from an ontology to reduce the dramatic growth in size of its SAT problem during model finding with increasing domain sizes. In the following two chapters, we will use ODE as FOL preprocessing technique to simplify ontologies and curb the otherwise very quick growth in the size of their SAT translations with increasing sized datasets and subsequently verify their improved model finding in practice.

CHAPTER 6

THE IMPACT OF ODE ON THE SIZE OF THE SAT PROBLEM FOR FOL MODEL FINDING

In Chapter 5 we identified two measures that – independent of a particular model finder — have an outsized impact on the size of the SAT translations of data-integrated FOL ontologies. They are: (1) the number of predicates of highest arity in the ontology, (2) the domain size of the ABox, i.e. the number of distinct named entities. Using examples we illustrated that the search space of the SAT problem determined by the number of propositional variables is exponential in the domain size of the dataset and the number of distinct predicates in an ontology, but double exponential in the highest arity of these predicates. The great majority of domain and application ontologies use unary and binary predicates (classes and relations) – in fact the language of more restricted ontology languages (DL-based, like OWL) is limited to those. However, when FOL ontologies are translated to SAT, existentially-quantified variables get skolemized introducting additional, mostly binary and ternary (and sometimes higher arity) predicates. This increase in signature during clausification of the ontology to FOL-CNF negatively influences solver performance. To overcome this drawback, we introduced in Chapter 4 P_v and P_c of the SAT representation as quantitative measures contributing to the hardness of model finding and *Optional Definition Elimination* (ODE) as an FOL formula simplification technique for specifically lowering P_v. In this chapter we address objective 3 (O3 in Section 1.2.2) of this dissertation to understand how specific size measures have the greatest impact on the hardness of model finding from a theoretical perspective. We study in more detail how ODE affects the size of the SAT problems resulting from different sized data-integrated ontologies. ODE reduces P_v by removing defined predicates of highest arity from an ontology. This reduction is performed before the FOL formula is clausified, and when judiciously applied to an ontology leads to a smaller SAT problem (with fewer clauses and variables) without changing its satisfiability and semantic meaning.

Through the systematic construction of different versions for a set of three sample ontologies, we analyze how removing different definition sets and replacing ground facts with defined assertions in the ABox correlate with the size of the resulting SAT problem. We hypothesize that in most cases, aggressive ODE on predicates of highest arity will yield a significant reduction in the number of propositional variables, but this reduction may sometimes be coupled with an increase in propositional clauses depending on the nature of formalization of the eliminated predicates. We present the theoretical implications of this assumption on the constructed sample ontologies by specifically trying to answer the following question: how does the elimination of optional predicates from an FOL ontology $\mathcal{O}$ have any bearing on the size of the resulting SAT problem as measured in terms of the three identified parameters[1] : the number of propositional variables, number of propositional clauses in the SAT problem $\mathcal{O}_{\text{CNF-d}}$, and (maximal and median)-width of the intermediary clausified formula $\mathcal{O}_{\text{FOL-CNF}}$.

6.1 Design of Study

Our own experience tells us that there is a lot of variability in the performance of model finders with FOL ontologies that arise from seemingly minor syntactic differences (names of relations, style of writing axioms, inclusion or exclusion of lemmas, etc.). To eliminate such factors, for each ontology (CODI, RCC and INCH) we construct sets of equivalent axiomatizations that differ in the inclusion or exclusion of additional definitions and the substitution of ground facts by defined assertions to keep the number of possible models constant regardless of whether extra definitions are present or not[2] . Optional Definition Elimination as introduced in Section 5.3 allows the removal of sets of definitions from the DBox of an ontology (with or without an ABox) without altering its semantic meaning. Using

[1]Many previous works focus on clauses-to-variables ratio as an indicator of the complexity or hardness of the problem [153, 51, 206, 247], but here we study the implications of their absolute values besides other measures.

[2]The number of models can be thought of as a hardness criteria as more models increase the chance to encounter a model early during the SAT solving process, thus leading to faster runtimes on average.

this technique we can reduce the number of propositional variables and clauses in its SAT translation – $\mathcal{O}_{\text{CNF-d}}$, for more efficient model finding.

6.1.1 Construction of TBoxes with Different Extents of ODE

In order to construct sets of ontologies (TBox and ABox) that admit equivalent models (apart from the defined predicates that can be reconstructed), we first construct sub-TBoxes for each of the three spatial ontologies: CODI, RCC and INCH. These ontologies are ideal for our study because: (1) they are about the right extent in terms of their size as measured in terms of the length and number of variables in the generated FOL-CNF clauses, (2) they have sets of binary defined predicates available for ODE, (3) model finding using them is difficult making them effective for our studies in understanding their hardness, (4) real datasets are readily available for these qualitative spatial ontologies – any spatial dataset from GIS can be accessed using terminology from the SFA-FOL formalization provided in Chapter 4. A secondary reason is that we simultaneously verify these ontologies[3] against real datasets, thus improving our confidence in the ontologies themselves and testing the feasibility of joint qualitative-geometric spatial reasoning as outlined in [256]. The different TBoxes that we construct – which we refer to as *cases* – differ only in the inclusion or exclusion of one or more definitions from its DBox. Details of the definitions included in each TBox is provided in Tables 6.1.2, 6.1.2 and **??**. For each theory we have a default case, which takes the original unaltered axiomatization of the theory (case 13, 7, and 4 for CODI, RCC, and INCH, respectively that contain $(|\Omega_{a=1}|, |\Omega_{a=2}|)$[4] = (8,13), (0,6), (0,7) predicates). In addition, we remove one or multiple definitions of binary predicates at a time[5] , resulting in a total of 13/7/4 cases for the three ontologies, with case 1 being the TBox with the least definitions included $((|\Omega_{a=1}|, |\Omega_{a=2}|) =$ (8,8), (0,1), (0,5) predicates for case 1 in each of the theories). The definitions that we chose for elimination are only some of the *optional definitions*. For example, case 1 for CODI still contains some defined predicates, as well as the ontologies'

[3]External verification of SFA-FOL is also made possible through a similar process.
[4]Number of unary and binary predicates in each theory.
[5]The terminologies of the studied ontologies primarily consist of binary predicates, but also some unary predicates. Only definitions for binary predicates are removed.

primitive predicates[6] . Table 6.1 provides the list of primitives, defined predicates (some are optional) that are not touched during ODE, and the optional predicates that are removed in some of the TBoxes for the three ontologies.

Simplification using ODE is expected to be most relevant to ontologies that meet the following necessary requirements: (1) have many explicit definitions, i.e. with a large DBox, (2) the explicit definitions build on top of each other and do not contain cyclic dependencies[7] , or taxonomic hierarchies.

Figure 6.1 shows the dependency graphs between predicates in CODI, RCC, and INCH ontologies, e.g. *PP* in CODI is defined using *P* in the definiens, but *P* itself is defined in terms of two primitives (*Cont* and *EqDim*). Then a simple ground fact using *PP* can recursively undergo ODE as follows:

Original sentence	*PP*(‘*segment1103*’, ‘*road_I95*’)
After removing *PP*	*P*(‘*segment1103*’, ‘*road_I95*’) $\wedge$ (*segment1103*’ $\neq$ ‘*road_I95*’)
After removing *P*	*Cont*(‘*segment1103*’, ‘*road_I95*’) $\wedge$ *EqDim*(‘*segment1103*’, ‘*road_I95*’) $\wedge$ (*segment1103*’ $\neq$ ‘*road_I95*’)

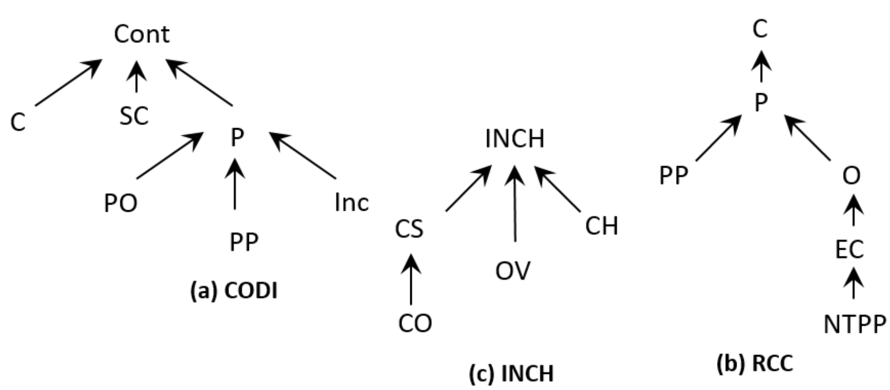


Figure 6.1: Dependencies between defined predicates in the RCC, CODI and INCH ontologies. They show the recursive structure of the defined predicates. For example, in CODI, *PP* is recursively defined using *Cont*.

[6]The primitives in an ontology are not defined and not available for ODE.
[7]For example, CODI, RCC, INCH, non-cyclic nature of dependencies, as observed from Figure 6.1

CODI	
Terms included in all cases	
Primitive binary terms	Cont, Leq
Primitive unary terms	S, ZEX
Defined binary terms	Lt, Gt, Geq, EqDim, Covers, P, PP, C, PO, Inc, SC
Defined unary terms	MinDim, MaxDim, PointRegion, Point, Curve, ArealRegion
Optionally defined terms that are removed in some cases	
PP(x,y) $\leftrightarrow$	$P(x, y) \wedge x \neq y$
C(x,y) $\leftrightarrow$	$\exists z[Cont(z, x) \wedge Cont(z, y)]$
PO(x,y) $\leftrightarrow$	$\exists z[P(z, x) \wedge P(z, y)]$
Inc(x,y) $\leftrightarrow$	$\exists z[Lt(z, x) \wedge Cont(z, x) \wedge P(z, y)] \vee \exists z[Lt(z, x) \wedge Cont(z, x) \wedge P(z, y)]$
SC(x,y) $\leftrightarrow$	$\exists z[Cont(z, x) \wedge Cont(z, y)] \wedge \forall z[Cont(z, x) \wedge Cont(z, y) \rightarrow Lt(z, x) \wedge Lt(z, y)]$
RCC	
Terms included in all cases	
Primitive binary terms	C, PP, O, EC, NTTP
Defined binary terms	P, PP, O, EC, NTPP
Optionally defined terms that are removed in some cases	
P(x,y) $\leftrightarrow$	$\forall z[C(z, x) \rightarrow C(z, y)]$
PP(x,y) $\leftrightarrow$	$P(x, y) \wedge \neg P(y, x)$
O(x,y) $\leftrightarrow$	$\exists z[P(z, x) \wedge P(z, y)]$
EC(x,y) $\leftrightarrow$	$C(x, y) \wedge \neg O(x, y)$
NTTP(x,y) $\leftrightarrow$	$PP(x, y) \wedge \neg \exists z[EC(z, y) \wedge EC(z, y)]$
INCH	
Terms included in all cases	
Primitive binary terms	INCH, GED
Primitive unary terms	ZEXI
Defined binary terms	CH, CS, CO, OV
Optionally defined terms that are removed in some cases	
CS(x,y) $\leftrightarrow$	$\forall z[INCH(x, z) \rightarrow INCH(y, z)]$
CH(x,y) $\leftrightarrow$	$INCH(x, y) \wedge \forall z[(INCH(x, z) \wedge INCH(z, x)) \rightarrow (INCH(y, z) \wedge INCH(z, y))]$
CO(x,y) $\leftrightarrow$	$\forall z[\neg ZEXI(z) \wedge CS(z, x) \wedge CS(z, y)]$
OV(x,y) $\leftrightarrow$	$\forall INCH(x, y) \wedge INCH(y, x)$

Table 6.1: Predicates (FOL literals) for each of the ontologies RCC, CODI and INCH used in the theoretical study here and empirical analysis in Chapter 7.

6.1.2 Constructing (r-d) ABoxes

The composition of ABoxes can vary widely: it may contain a handful or thousands of facts, and some predicates may be used much more than others. In the extreme case, many predicates may only rarely or not at all be used in an ABox. To study the impact of the ABox in a more systematic way, we need to carefully control its size and makeup. Thus we have designed the study to control two parameters: (1) the domain size d of a model which corresponds to the number of distinct spatial objects (i.e. *individuals* in ontology parlance) in a sample ABox, and (2) the assertion density r, which indicates how many assertions for each binary *optional* predicate in the default ontology are included. More precisely, for a given r, we aim to include the same number of (r) positive and (r) negative assertions for each binary predicate in the DBox. Such an ABox is called an (r-d)*ABox* defined as follows:

Definition 24. *Let $\mathcal{O}$ be an ontology and D a domain of individuals. ABox($\mathcal{O}$) is called a* (r-d)ABox *iff it contains the following assertions:*

1. *For each $\Omega \in \lambda(\mathcal{O})$ with arity $a(\Omega) \geq 2$, ABox($\mathcal{O}$) contains exactly r ground positive assertions (i.e. of the form $\Omega(d_1, d_2, \dots)$) and exactly r ground negated assertions (i.e. of the form $\neg\Omega(d'_1, d'_2, \dots)$ where $d_i, d'_i \in D$;*
2. *ABox($\mathcal{O}$) contains at most one sentence of the format $\Omega(d)$ for each $d \in D$ where Ω is a unary predicate (i.e. $\Omega \in \lambda(\mathcal{O})$ and $a(\Omega) = 1$)[8] ;*
3. *Distinctness assertions of the form $d_i \neq d_j \in$ABox($\mathcal{O}$) for each pair $(d_i, d_j) \in D$ with $d_i \neq d_j$.*

[8]This criteria captures the idea that each individual in the domain can be asserted to be a member of some class; but this restriction does not significantly impact the overall size of the ABox or the resulting SAT problem, which is dominated by the number of assertions of the first kind.)

		TBox												Basic ABox (i.e. r = 1)					
Case	**Defined Predicates included**	$\lvert\Omega_{a=1}\rvert$	$\lvert\Omega_{a=2}\rvert$	$\lvert C_T\rvert$	$\lvert C_T\rvert$ with v FOL variables				$\lvert C_T\rvert$ with $W \geq 3$	$\lvert sf_{a=1}\rvert$	$\lvert sf_{a=2}\rvert$	$\lvert sf_{a=3}\rvert$	$\lvert C_A\rvert$	$\lvert C_A\rvert$ with v variables		$\lvert C_A\rvert$ with width w		$\lvert sf_{a=1}\rvert$
					v=3	v=2	v=1	v=0	-					v=1	v=0	w=4	w=3	
1	- (all cases include 22 other predicates)	8	8	65	3	32	29	1	31	1	5	1	27	10	17	1	6	6
2	PP	8	9	68	3	35	29	1	33	1	5	1	26	9	17	1	6	6
3	C	8	9	68	4	34	29	1	33	1	5	2	25	8	17	1	6	5
4	C + PP	8	10	71	4	37	29	1	35	1	5	2	25	8	17	1	6	5
5	PO	8	9	68	4	34	29	1	33	1	5	2	26	8	18	1	6	5
6	PO + PP	8	10	71	4	37	29	1	33	1	5	2	25	8	17	1	6	5
7	PO + PP + C	8	11	74	5	39	29	1	35	1	5	3	24	7	17	1	6	4
8	Inc	8	9	76	5	41	29	1	43	1	5	3	18	7	11	1	4	4
9	Inc + PP	8	10	79	5	44	29	1	43	1	5	3	17	7	10	1	4	4
10	Inc + PP + C + PO	8	12	85	7	48	29	1	45	1	6	4	15	5	10	1	4	2
11	SC	8	9	72	8	34	29	1	36	1	5	3	22	4	18	0	2	4
12	SC + PP	8	10	75	8	37	29	1	38	1	5	3	21	4	17	0	2	4
13	SC + PP + C + PO + Inc	8	13	92	12	50	29	1	51	1	5	7	10	0	10	0	0	0

Table 6.2: Quantitative summary of the TBoxes, the FOL-CNF formulas of these TBoxes, and the basic ABox for the 13 cases experimented with in **CODI**. Each row represents one case, indicating the included optional definitions, and statistics of the resulting FOL-CNF ontologies. The abbreviations denote: $\Omega_{a=2}, \Omega_{a=1}$: binary and unary predicates; C: FOL-CNF clauses; v: variables in a FOL-CNF clause; w: literals in a FOL-CNF clause; $sf_{\mathrm{a}=1}, sf_{\mathrm{b}}$ - unary and binary skolem functions introduced in the conversion to FOL-CNF; $|C_T|$: number of FOL-CNF clauses from the TBox; $|C_A|$: number of FOL-CNF clauses from a Basic ABox (that is, for an ABox with $r = 1$).

6.2 The Impact of ODE on the Size of the SAT Problem

In this section we investigate the following dependencies between specifications of a $\mathcal{O}_{\text{CNF-d}}$ problem: (1) variation in propositional variables (P_v) and propositional clauses (P_c) with increasing use of ODE, (2) variation in P_v and P_c with increasing domain size and number of relational assertions – P_v vs. d, r and P_c vs. d, r.

Graph 6.3 shows the trends for P_v and P_c for CODI, RCC and INCH across the cases when ODE is applied at various degrees (various sets of defined predicates being removed) for increasing domain sizes d or increasing r values. The graphs clearly show that P_v increases polynomially with increasing d, while r has a lesser impact. These changes are analyzed further in Section 6.2.1. But the differences between the cases in the graphs also show that P_c also significantly grows with an increasing number of predicates in the TBox as further analyzed in Sec 6.2.2. P_v and P_c for different (r-d) values are calculated from size measures of the clausified TBox and a basic ABox[9] .

Building on Lemmas 1 and 2, the size of the SAT problem resulting from an (r-d)ABox can now be calculated as follows:

Lemma 3. *Let $\mathcal{O}$ be an FOL ontology with ABox($\mathcal{O}$) being an (r-d)ABox thereof. $|\Omega_{a=i}|$ is the set of predicates in $\mathcal{O}_{FOL\text{-}CNF}$ with maximum arity denoted by a^*. Let v^* be the maximum number of FOL variables in a single clause in $\mathcal{O}_{FOL\text{-}CNF}$.*
Then the resulting propositional SAT problem contains

- $P_v = \sum_{i=1}^{a^*} d^i \cdot |\Omega_{a=i}| + r \cdot \sum_{i=1}^{a^*} d^i \cdot |sf_{A,a=i}|$ *propositional variables; and*
- $P_c = \sum_{i=0}^{v^*} d^i \cdot |C_{T,v=i}| + r \cdot \sum_{i=0}^{v^*} d^i \cdot |C_{A,v=i}|)$ *propositional clauses.*

The last terms in each of these formulas capture the ABox's contribution – in terms of the number of assertions – to the size of the SAT problem. But it becomes clear that for factual ABoxes (and without any definition elimination), this contribution is negligible: P_v

[9] An ABox that contains exactly one positive and one negated assertion for each of the optionally defined terms in the theory.

		ABox for r = 5 (Total of 40 relational assertions)								ABox for r = 10 (Total of 80 relational assertions)							
S.No	**Defined predicates included**	P_v				P_c				P_v				P_c			
		d=20	d=30	d=40	d=50	d=20	d=30	d=40	d=50	d=20	d=30	d=40	d=50	d=20	d=30	d=40	d=50
1	- (all cases include 22 other predicates)	13980	39870	86360	159450	37906	111416	245326	457636	14580	40770	87560	160950	38991	113001	247411	460221
2	PP	14380	40770	87960	161950	39006	113966	249926	464886	14980	41670	89160	163450	39991	115401	251811	467221
3	C	22280	67620	151760	286700	46506	139916	312126	587136	22780	68370	152760	287950	47391	141201	313811	589221
4	C + PP	22680	68520	153360	289200	47706	142616	316926	594636	23180	69270	154360	290450	48591	143901	318611	596721
5	PO	22280	67620	151760	286700	46511	139921	312131	587141	22780	68370	152760	287950	47401	141211	313821	589231
6	PO + PP	22680	68520	153360	289200	47706	142616	316926	594636	23180	69270	154360	290450	48591	143901	318611	596721
7	PO + PP + C	30980	96270	218760	416450	56406	171266	383926	724386	31380	96870	219560	417450	57191	172401	385411	726221
8	Inc	30180	94470	215560	411450	57176	173036	387096	729356	30580	95070	216360	412450	57931	174141	388551	731161
9	Inc + PP	30580	95370	217160	413950	58371	175731	391891	736851	30980	95970	217960	414950	59121	176831	393341	738651
10	Inc + PP + C + PO	39580	124770	285560	545950	75771	233031	525891	996351	39780	125070	285960	546450	76321	233831	526941	997651
11	SC	30180	94470	215560	411450	78111	247321	567331	1086141	30580	95070	216360	412450	78601	248011	568221	1087231
12	SC + PP	30580	95370	217160	413950	79306	250016	572126	1093636	30980	95970	217960	414950	79791	250701	573011	1094721
13	SC + PP + C + PO + Inc	63380	205470	477160	920450	116071	369081	848091	1625101	63380	205470	477160	920450	116121	369131	848141	1625151

Table 6.3: P_v and P_c in the propositional formulas for different ABox sizes for the 13 cases experimented within **CODI**. Each row represents one case, indicating the included optional definitions, and example statistics of the resulting propositionalized versions for samples sizes 20, 30, 40, and 50. d: domain size (i.e. distinct individuals in the ABox samples), and r values 5, 10.

		TBox							Basic ABox (i.e. r = 1)										
Case	**Defined predicates included**	$\lvert\Omega_{a=2}\rvert$	$\lvert C_T\rvert$	$\lvert C_T\rvert$ with v FOL variables			$\lvert C_T\rvert$ with $w \geq 3$	$\lvert sf_{a=3}\rvert$	$\lvert C_A\rvert$	$\lvert C_A\rvert$ with v variables				$\lvert C_A\rvert$ with width w				$\lvert sf_{a=1}\rvert$	$\lvert sf_{a=2}\rvert$
				v=3	v=2	v=1				v=3	v=2	v=1	v=0	w=6	w=5	w=4	w=3		
1	C	1	2	0	1	1	0	0	45	4	16	20	5	4	16	4	8	7	10
2	C+P	2	6	1	4	1	2	1	20	0	0	8	12	0	0	6	2	3	2
3	C+P+PP	3	9	1	7	1	3	1	18	0	0	8	10	0	0	4	2	3	2
4	C+P+O	3	9	2	6	1	3	2	15	0	0	1	14	0	0	1	4	1	0
5	C+P+PP+O	4	12	2	9	1	4	2	13	0	0	1	12	0	0	1	0	1	0
6	C+P+PP+O+EC	5	15	2	12	1	5	2	10	0	0	1	9	0	0	0	0	1	0
7	C+P+PP+O+EC+NTPP	6	19	3	11	1	8	2	8	0	0	0	8	0	0	0	0	0	0

		ABox for r = 5 (Total of 40 relational assertions)								**ABox for r = 10 (Total of 80 relational assertions)**							
S.No	**Defined predicates included**	P_v				P_c				P_v				P_c			
		d=20	d=30	d=40	d=50	d=20	d=30	d=40	d=50	d=20	d=30	d=40	d=50	d=20	d=30	d=40	d=50
1	C	21100	46950	83000	129250	194425	615925	1413625	2707525	41800	93000	164400	256000	388450	1230950	2825650	5412550
2	C+P	13100	38250	83800	155750	10460	31860	72060	137060	17400	47700	100400	181500	11320	33120	73720	139120
3	C+P+PP	13500	39150	85400	158250	11650	34550	76850	144550	17800	48600	102000	184000	12500	35800	78500	146600
4	C+P+O	17300	56850	133000	257750	18570	59620	137870	265320	17400	57000	133200	258000	18740	59840	138140	265640
5	C+P+PP+O	17700	57750	134600	260250	19760	62310	142660	272810	17800	57900	134800	260500	19920	62520	142920	273120
6	C+P+PP+O+EC	18100	58650	136200	262750	20945	64995	147445	280295	18200	58800	136400	263000	21090	65190	147690	280590
7	C+P+PP+O+EC+NTPP	26400	86400	201600	390000	28440	90940	209640	402540	26400	86400	201600	390000	28480	90980	209680	402580

Table 6.4: Quantitative summary of the TBoxes, FOL-CNF formulas of these TBoxes, the basic ABox, and (r-d)-ABoxes of the 7 cases experimented with in **RCC**. Each row represents one case, indicating the included optional definitions, and statistics of the resulting FOL-CNF ontologies.
The abbreviations denote: $\Omega_{a=2}, \Omega_{a=1}$: binary and unary predicates; C: FOL-CNF clauses; v: variables in a FOL-CNF clause; w: literals in a FOL-CNF clause; $sf_{\mathrm{a=1}}, sf_{\mathrm{b}}$ - unary and binary skolem functions introduced in the conversion to FOL-CNF; $|C_T|$: number of FOL-CNF clauses from the TBox; $|C_A|$: number of FOL-CNF clauses from a Basic ABox (that is, for an ABox with $r = 1$).

		TBox							Basic ABox (i.e. r = 1)									
Case	**Defined predicates included**	$\|\Omega_b\|$	$\|C_T\|$	$\|C_T\|$ with v FOL variables			$\|C_T\|$ with $w \geq 3$	$\|sf_{a=3}\|$	$\|C_A\|$	$\|C_A\|$ with v variables				$\|C_A\|$ with width w		$\|sf_{a=1}\|$	$\|sf_{a=2}\|$	$\|sf_{a=3}\|$
				v=3	v=2	v=1	-			v=3	v=2	v=1	v=0	w=4	w=3			
1	INCH+CS+CH	5	33	10	21	2	19	7	42	10	21	3	8	4	16	1	0	7
2	INCH+CS+CH+OV	6	36	10	24	2	20	7	44	10	24	3	7	4	17	1	0	7
3	INCH+CS+CH+CO	6	39	11	24	4	20	8	44	11	24	2	7	5	15	0	0	8
4	INCH+CS+CH+CO+OV	7	42	11	27	4	21	8	45	11	27	2	5	5	16	0	0	8

		ABox for r = 5 (Total of 20 relational assertions)								ABox for r = 10 (Total of 40 relational assertions)							
S.No	**Definitions**	P_v				P_c				P_v				P_c			
		d=20	d=30	d=40	d=50	d=20	d=30	d=40	d=50	d=20	d=30	d=40	d=50	d=20	d=30	d=40	d=50
1	INCH+CS+CH	58100	193650	456200	887750	88540	289090	673840	1302790	58200	193800	456400	888000	88680	289280	674080	1303080
2	INCH+CS+CH+OV	58500	194550	457800	890250	89735	291785	678635	1310285	58600	194700	458000	890500	89870	291970	678870	1310570
3	INCH+CS+CH+CO	66400	221400	521600	1015000	97635	318635	742435	1435035	66400	221400	521600	1015000	97670	318670	742470	1435070
4	INCH+CS+CH+CO+OV	66800	222300	523200	1017500	98830	321330	747230	1442530	66800	222300	523200	1017500	98860	321360	747260	1442560

Table 6.5: Quantitative summary of the TBoxes, FOL-CNF formulas of these TBoxes, the basic ABox, and (r-d)-ABoxes of the 4 cases experimented with in **INCH**. Each row represents one case, indicating the included optional definitions, and statistics of the resulting FOL-CNF ontologies.
The abbreviations denote: $\Omega_{a=2}, \Omega_{a=1}$: binary and unary predicates; C: FOL-CNF clauses; v: variables in a FOL-CNF clause; w: literals in a FOL-CNF clause; $sf_{\mathrm{a=1}}, sf_{\mathrm{b}}$ - unary and binary skolem functions introduced in the conversion to FOL-CNF; $|C_T|$: number of FOL-CNF clauses from the TBox; $|C_A|$: number of FOL-CNF clauses from a Basic ABox (that is, for an ABox with $r = 1$).

will not change at all (because ground unit clauses do not yield any Skolem functions), while P_c includes exactly as many extra clauses as are contained in the ABox. Even for ABoxes with thousands of facts, this is relatively small compared to the number of clauses that are generated from the TBox for growing domain sizes. This shows that the size of the ABox in terms of r is not really a problem for model finding, but the signature of the TBox and domain size are.

For example, P_v and P_c for the default cases for CODI, RCC, and INCH for domain size $d = 20$ and $r = 1$ are (26,400, 28,408), (63,380, 116,031), and (66,800, 98,806). For the same domain size, when $r = 20$, P_v and P_c are (26,400, 28,560), (63,380, 116,221), and (66,800, 98,920), but when the domain size is doubled (i.e. $d = 40$), they yield the following values: (201,600, 209,760), (477,160, 848,241), and (523,200, 747,320). This informs that any differences or significant increase in P_v and P_c (that will influence model finding) arises from the first terms in the formulas – the number of predicates in the TBox and their arity, and domain size.

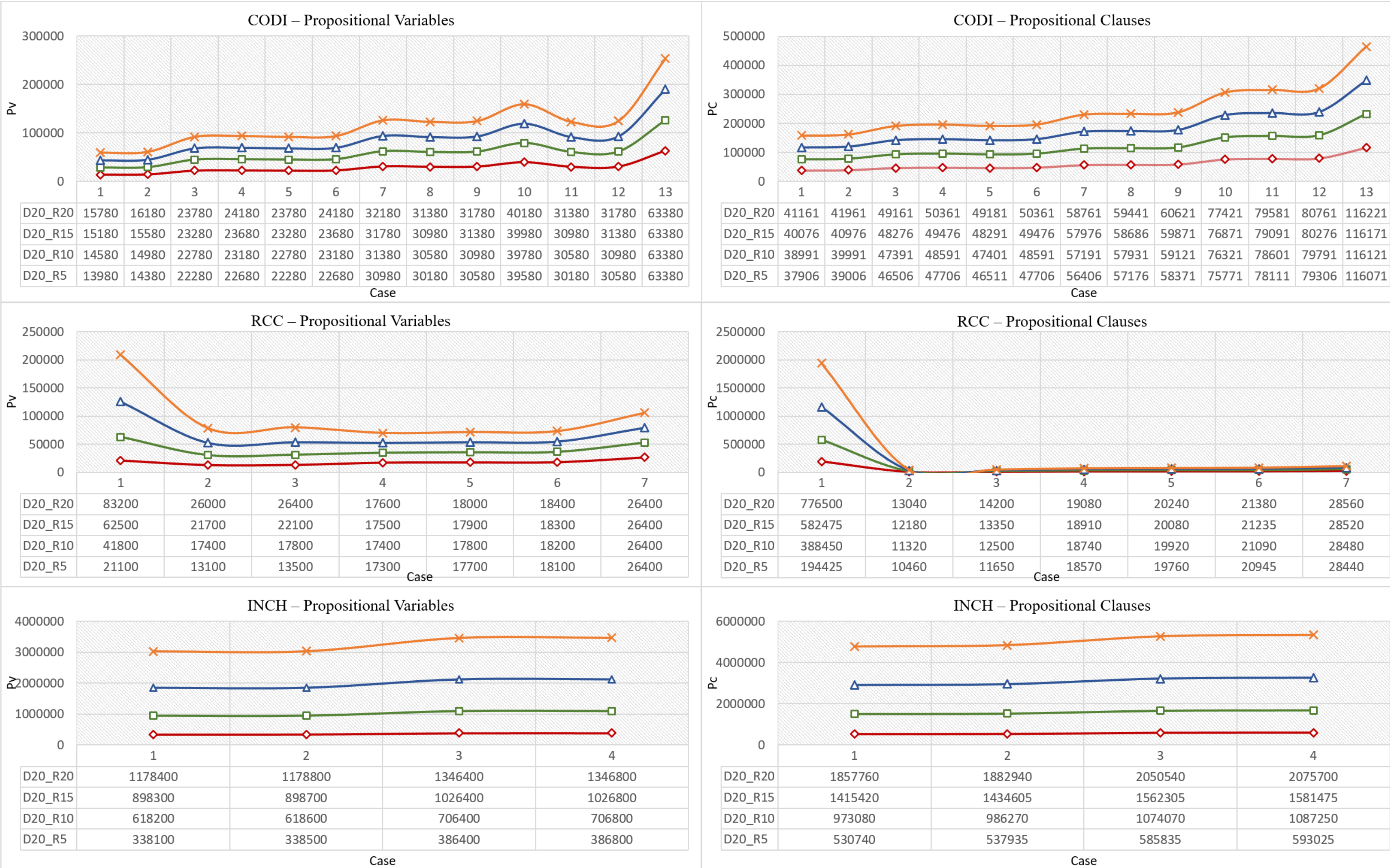
6.2.1 Growth in Propositional Variables with Different (r-d)ABoxes and Different Definition Sets

Reiterating from Chapter 5, the search for a model for $\mathcal{O}_{\text{CNF-}d}$ has the worst-case complexity $O(P_v) = O(|\Omega_{a=a^*}| \cdot d^{a^*})$ where a^* is the highest arity of all predicates in $\mathcal{O}_{\text{FOL-CNF}}$. This search space, which is set by P_v is exponential in the size of the terminology of the ontology.

Influence of ODE (with different sets of eliminated definitions): Overall, P_v decreases with an increased number of definitions being removed, though the reduction is minimal in some cases (e.g. removing the comparatively simple definition of PP from CODI's case 2 decreases P_v only between 1-4% for different r-d values), and sometimes the decrease is substantial when the removed definition is longer or more complex (e.g. removing SC from CODI's case 12 decreases P_v between 50-60% for different r-d values). The elimination of the five binary predicates SC, Inc, PO, PP and C from CODI (from case 13 to case 1) reduces the number of propositional variables to roughly one-third even though 9 other binary predicates are still maintained (i.e. only 67% of all predicates are kept). Then there are the cases where elimination of nested-defined predicates (e.g. a definition using in its definiens other defined predicates slated for elimination) leads to the addition of Skolem functions[10] that get translated to predicates of higher arity. This is seen in RCC's case 1, whose DBox has 0 optional predicates (eliminating the five optional predicates P, PP, O, EC and $NTPP$), however clausification results in a larger signature mostly contributed from the ABox (7 unary and 10 binary Skolem functions are added from a basic ABox) leading to very large values for P_v. But cases 2 and 4 in RCC demonstrate a significant decrease from the default case, where the removal of 4 and 3 optional predicates (but not removing P and O), respectively, decreases P_v from 390,000 (case 7) to 207,250 (case 2) and 209,750(case 3), approximately 47% decrease for $d = 50$ and $r = 15$. The trends of P_v for INCH seem relatively flat, because any decrease in signature from ODE gets counteracted by new predicates being added as the result of skolemization of the TBox and ABox. Still the removal of even a single definition moderately reduces P_v, although the values still remain very large even for very small r-d values. For $d = 20$ and $r = 5$, the removal of two binary predicates CO and OV in INCH (from case 4 to case 1) reduces P_v by 13% from 66,800 to 58,100.

Influence of domain size: P_v is only really dependent on the number of distinct predicates in an ontology, their arity, and d. In general for CODI, RCC, and INCH P_v increases polynomially (in d^3 since $a^* = 3$ for all three ontologies) with increasing domain size for any case. For example, in CODI, P_v for (case 1, case 13) (i.e. the minimal case containing 8 unary + 8 binary predicates and the default case containing 8 unary + 13 binary predicates) is (13,980, 63,380), (39,870, 205,470), (86,360, 477,160), (159,450, 920,450) for $d = 20$, 30, 40 and 50 respectively (for $r = 5$, cf. Table 6.1.2).

[10]Skolem constants and Skolem functions operate just like any other FOL predicate of the next higher arity.



CODI – Propositional Variables

	1	2	3	4	5	6	7	8	9	10	11	12	13
D20_R20	15780	16180	23780	24180	23780	24180	32180	31380	31780	40180	31380	31780	63380
D20_R15	15180	15580	23280	23680	23280	23680	31780	30980	31380	39980	30980	31380	63380
D20_R10	14580	14980	22780	23180	22780	23180	31380	30580	30980	39780	30580	30980	63380
D20_R5	13980	14380	22280	22680	22280	22680	30980	30180	30580	39580	30180	30580	63380

CODI – Propositional Clauses

	1	2	3	4	5	6	7	8	9	10	11	12	13
D20_R20	41161	41961	49161	50361	49181	50361	58761	59441	60621	77421	79581	80761	116221
D20_R15	40076	40976	48276	49476	48291	49476	57976	58686	59871	76871	79091	80276	116171
D20_R10	38991	39991	47391	48591	47401	48591	57191	57931	59121	76321	78601	79791	116121
D20_R5	37906	39006	46506	47706	46511	47706	56406	57176	58371	75771	78111	79306	116071

RCC – Propositional Variables

	1	2	3	4	5	6	7
D20_R20	83200	26000	26400	17600	18000	18400	26400
D20_R15	62500	21700	22100	17500	17900	18300	26400
D20_R10	41800	17400	17800	17400	17800	18200	26400
D20_R5	21100	13100	13500	17300	17700	18100	26400

RCC – Propositional Clauses

	1	2	3	4	5	6	7
D20_R20	776500	13040	14200	19080	20240	21380	28560
D20_R15	582475	12180	13350	18910	20080	21235	28520
D20_R10	388450	11320	12500	18740	19920	21090	28480
D20_R5	194425	10460	11650	18570	19760	20945	28440

INCH – Propositional Variables

	1	2	3	4
D20_R20	1178400	1178800	1346400	1346800
D20_R15	898300	898700	1026400	1026800
D20_R10	618200	618600	706400	706800
D20_R5	338100	338500	386400	386800

INCH – Propositional Clauses

	1	2	3	4
D20_R20	1857760	1882940	2050540	2075700
D20_R15	1415420	1434605	1562305	1581475
D20_R10	973080	986270	1074070	1087250
D20_R5	530740	537935	585835	593025

Figure 6.2: Variation in P_v (graphs on the left) and P_c (graphs on the right) with increasing r for constant domain size, $d = 20$, for each of the ontologies, RCC, CODI, and INCH calculus.

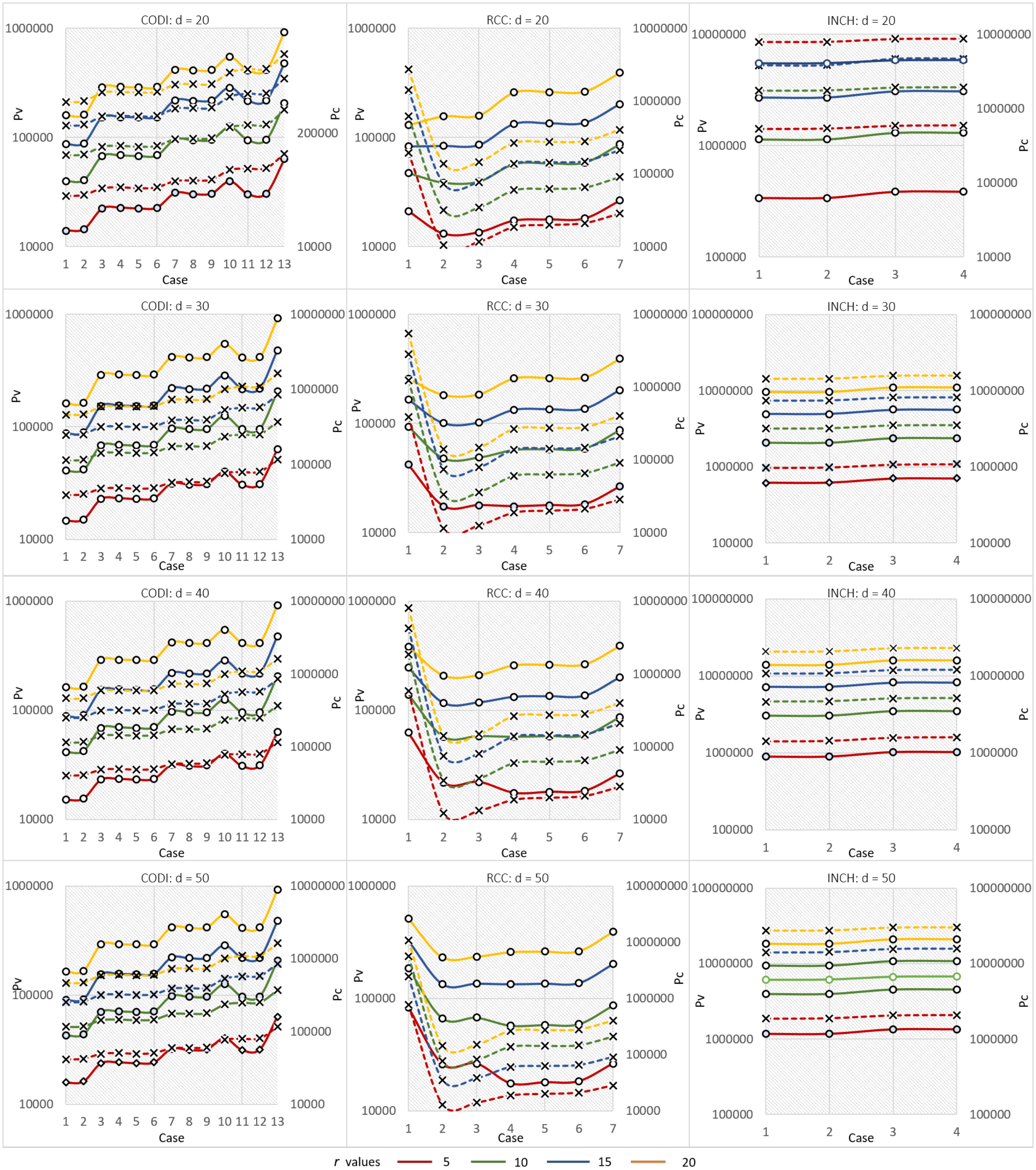


Figure 6.3: Number of P_v and P_c in preprocessed $\mathcal{O}_{\text{FOL-CNF}}$ formulas (for RCC, CODI and INCH) including a TBox and ABox starting from domain size 20 to 50 in increments of 10, and r ranging from 5 to 15 in increments of 5. The horizontal axis represents cases for different definition sets in the ontology. The primary vertical axis gives the absolute number of P_v (*solid lines*) and the secondary vertical axis gives the absolute number of P_c (*dashed lines*) in each CNF formula on a log_{10} scale.

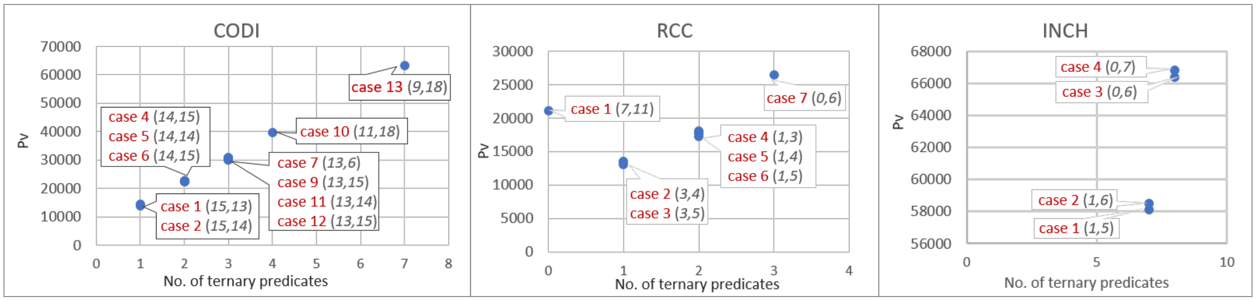


Figure 6.4: Graph showing the variation of P_v for each case with increasing size of terminology in an ontology with $d = 20$, $r = 5$ for CODI, RCC and INCH. The x-axis is the number of ternary predicates, the y-axis P_v, and the cases are shown as dots along with the number of unary and binary predicates included in them. The number of predicates are from the clausification of the TBox and a basic ABox (i.e. for $r = 1$). Intuitively, points closer to the origin are the ones that are deemed the easiest.

P_v increases between 6 - 8 folds for both case 1 and case 13 when d doubles (i.e. going from $d = 20$ to $d = 40$). Similarly, in RCC for case 7 containing 6 binary predicates, when $r = 5$, P_v increases from 26,400 for $d = 20$ to 201,600 for $d = 40$. This is an increase of atleast one order of magnitude. INCH ontology shows a quicker, almost exponential growth in P_v with increasing d, and this is due to the presence of many additional ternary predicates from skolemization. The decrease in number of ternary predicates in INCH is minimal post-ODE (8 in case 4 to 7 in case 1), and therefore even with the empty DBox, P_v is still large (as compared to CODI and RCC with similar domain sizes) for the smallest r-d values tested.

Influence of number of relational assertions: Unlike for d, there does not seem to be a similar exponential growth in P_v with increasing r for any of the three ontologies. For instance Graph 6.2 shows P_v growing slowly for CODI, by 4%, for r ranging from 5 to 20 in increments of 5, for a constant domain size $d = 20$. Any change of P_v across r for a case depends on any additional predicates included in the problem from the skolemization of assertions in the ABox. This occurs in cases where we remove predicates during ODE, thereby replacing assertions with sentences that may contain existential quantifiers whose variables are bound by universal quantifiers. All three ontologies introduce Skolem functions in their ABoxes, but while CODI only introduces unary functions (column 19 in Table 6.1.2), RCC introduces binary (column 20 in Table 6.1.2) and INCH introduces ternary predicates (column 19 in Table 6.1.2). This addition influences the comparatively quicker growth of P_v across r for RCC and INCH (cf. Graph 6.2). For example in RCC, the removal of the predicate O in case 2 adds two new unary Skolem functions in the basic ABox (i.e. for $r = 1$) – and therefore two additional binary predicates in the ontology. When the domain size increases from 40 to 50 (a 25% increase), with low r values ($r = 5$), P_v increases by 85%, but with larger r ($r = 20$), P_v almost doubles (increases by 98%). This increase although not exponential still significantly contributes to problem hardness. Cases - 1,2,3 in RCC introduce (10,2,2) binary predicates in their basic ABoxes. The large number of binary predicates in RCC's case 1 leads to the peaking of P_v, which is also noticeable by its significantly increasing values across r in Graph 6.2 - RCC-P_v. In INCH, cases - 1,2 add 7 ternary predicates and cases - 3,4 add 8 ternary predicates in their basic ABoxes. These cases in RCC and INCH are the ones whose P_v values are significantly affected by r.

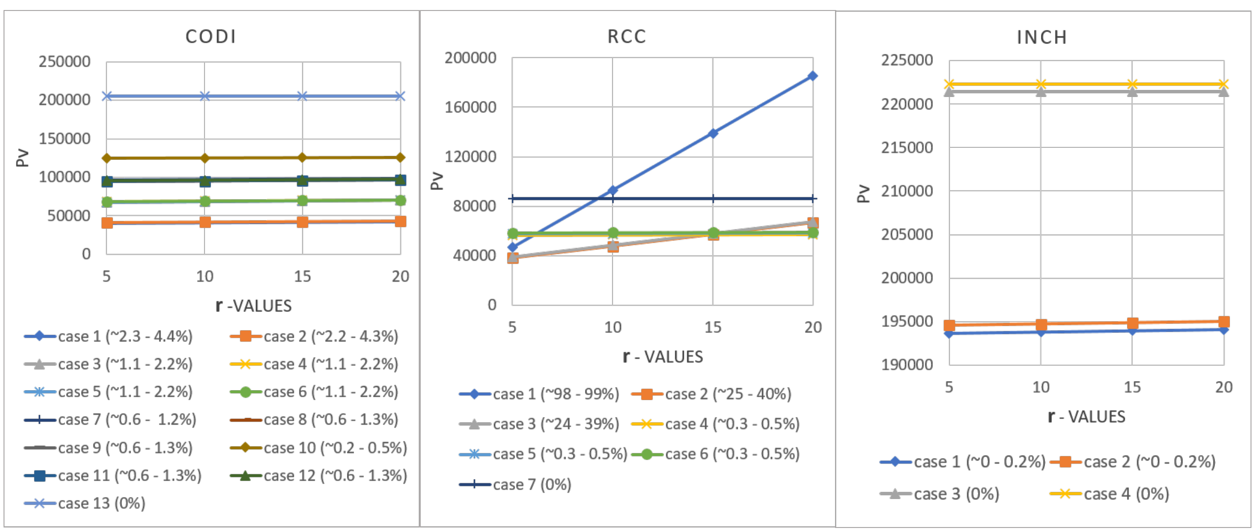


Figure 6.5: Graph plotting P_v (y-axis) against r (x-axis) for this $d = 30$. The legend shows the % increase in P_v when r doubles, i.e. going from $r = 5$ to $r = 10$, and then from $r = 10$ to $r = 20$.

The analysis of the growth in P_v shows that the feasibility of ontology verification against data is constrained by a large signature in the TBox (mostly binary defined predicates), and an increasing domain size. But eliminating appropriate definitions from an ontology inhibits this exponential growth in P_v, which will allow us to reason over ontologies with larger datasets.

6.2.2 Growth in Propositional Clauses with Different (r-d)ABoxes and Different Definition Sets

In Chapter 5 we have hypothesized that scalability of model finding also depends on the number of propositional clauses and (median) width of FOL-CNF clauses, as it determines how quickly the saturation algorithm terminates. Lemma 3 shows that P_c is influenced by the number of FOL-CNF clauses from the TBox and ABox and the domain size. Now we will take a closer look at the growth in P_c by studying ontologies constructed from different TBoxes and $(r\text{-}d)$ABoxes.

Influence of ODE (with different sets of eliminated definitions): P_c is polynomial in the number of FOL-CNF clauses, and exponential in the highest number of FOL variables in any clause in the formula, given by v^*. All cases in the three ontologies have formulas with an average of 2 variables in their FOL-CNF clause set, but even a single clause with 3 variables increases P_c significantly. Graph 6.6 shows that DBoxes with more optional definitions have v^* (max. number of variables) at most 3. Moreover case 7 in RCC, cases 10-12 in CODI, and cases 3-4 in INCH have more clauses with width ≥ 3. ODE reduces the number of FOL-CNF clauses in the TBox. Though with increasing number of definitions being eliminated, the ABox is no more factual i.e, a set of ground clauses, but has longer FOL sentences that produce more FOL-CNF clauses (high formula-length). This increase is much greater when the degree of nesting of predicates in definiens sentences in the ABox post-ODE is high, e.g. in RCC, with a default ABox, the number of FOL-CNF clauses in case 7 (having only ground clauses) is only 8, whereas case 1 (having at least two sentences with 6 universally quantified and 3 existentially quantified sub-formulas) is 45. P_c for case 1 in RCC is thus exceedingly high. And therefore the most eager definition elimination that is theoretically possible is not always necessarily desirable. In CODI and INCH any increase in FOL-CNF clauses in the ABox resulting from ODE is mostly counteracted by a decrease in clauses in the TBox. But generally ODE still decreases P_c, as it results in formulas with a lower v^* (≤ 2). For example, in CODI, going from case 13 to case 1, the number of clauses with $v \geq 3$ decreases from 12 to 3, leading to a reduction of P_c from 1,625,101 to 457,636 for $d = 50$ and $r = 5$ – amounts to one order of magnitude reduction.

Influence of domain size: Similar to P_v, P_c also increases polynomial with increasing domain size (polynomial in d^3 since $a^* = 3$ for all three ontologies), but does not grow similar to P_v w.r.t number of predicates in the ontology, rather more P_c depends on the length and complexity of the sentences of these predicates. In all the three ontologies, P_c increases proportional to P_v (since both the highest arity - a^*, and highest variable-density - v^* are 3), the growth is more with lower domain sizes (e.g. in CODI P_c increases by 3 times when going from $d = 20$ to $d = 30$, while P_c doubles when going from $d = 40$ to $d = 50$). Case 1 in RCC is the exception, where the ontology has a substantially larger clause set (a larger number of clauses is somewhat expected when replacing definitions with their definiens) while the number of propositional variables decreases. For example, when $r = 5$, (P_v, P_c) for case 1 (0 optional predicates) and case 7 (5 optional predicates) take values (21,100, 194,425) and (26,400, 28,440) respectively for $d = 20$, and (129,250, 2,707,525) and (390,000, 402,540) respectively for $d = 50$. In other words when d doubles (going from d = 20 to 40), in case 7 (without ODE), there is a 7 times increase in both P_v and P_c, whereas in case 1 (removing 5 definitions), P_v only increases 4-fold, but P_c increases 7-fold. Removing the 5 optional predicates in RCC (case 1) results in 22% and 66% decrease in P_v for $d = 20$, 50, but also leads to a 7-fold increase in P_c (the pattern of growth/reduction in the number of variables and clauses is the same across domain sizes). In this situation, it is ideal to remove some but not all optional predicates from the formula, since the goal is to reduce P_v while also avoiding an explosion of P_c[11] , which happens with case 2 in RCC, with 3-times reduction in P_c across increasing domain sizes.

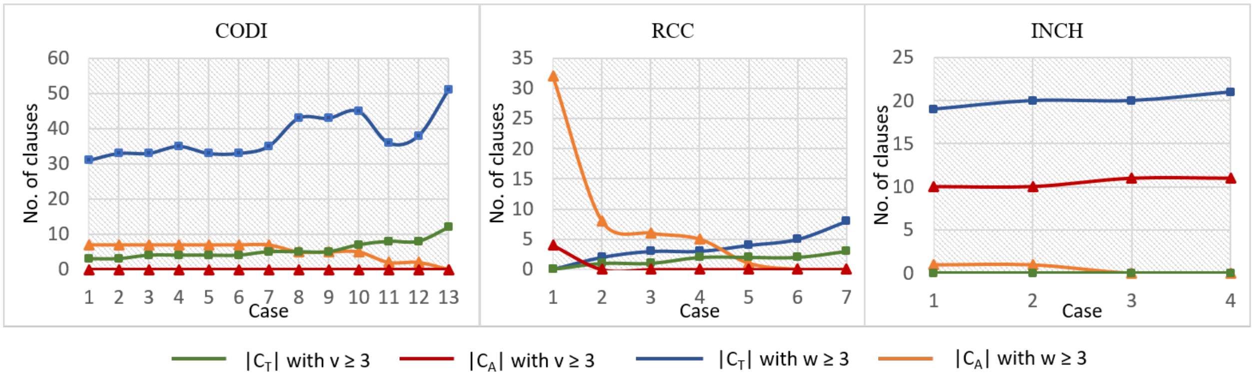


Figure 6.6: Graphs indicating number of clauses with three or more FOL variables (i.e. $v \geq 3$), and with three or more FOL literals (i.e. $w \geq 3$) in the FOL-CNF representations for CODI, RCC and INCH. $|C_T|$ and $|C_A|$ represent the number of clauses from the TBox and a basic ABox ($r = 1$) respectively. Numbers for $|C_A|$ increases with increasing r-values.

Influence of number of relational assertions: For a specific domain size, P_c increases polynomial with respect to v^*, and linearly with r – but by a small factor – for example in INCH starting with $r = 5$, as r doubles, the % increase in P_v doubles but minimally in both the default case (0.03%) and case 1 (0.15%). In CODI, although P_v remains constant, P_c increases with growing r across d, but this growth is still very minimal. For $d = 20$, in case 1, P_c increases by 3% when we go from $r = 5$ to 10, and for case 13 this increase is negligible (∼0.04%). For the RCC ontology P_c has a growth pattern similar to P_v. When r increases from 5 to 10, P_v doubles for case 1 (for $d = 20$ from 21,100 to 41,800), and P_c increases by a similar amount (for $d = 20$ from 194,425 to 388,450), whereas in case 7, where P_v remains unchanged P_c grows only by a trivial number (∼0.1%). This is revealed in Lemma 3, i.e. the significant growth of P_c is due to the effect of ODE on the ABox that results in a longer FOL-CNF formula – FOL-CNF formulas for case 1 (with aggressive ODE) has length = 47,

[11]Only the empirical study in the next section can give us more insights about which definitions should be replaced to obtain a optimal balance between reduction of the number of propositional variables and the addition of large numbers of clauses

whereas case 7 (no ODE) has length = 27. This is the same situation for the INCH ontology, where P_c grows gradually with increasing r for all four cases.

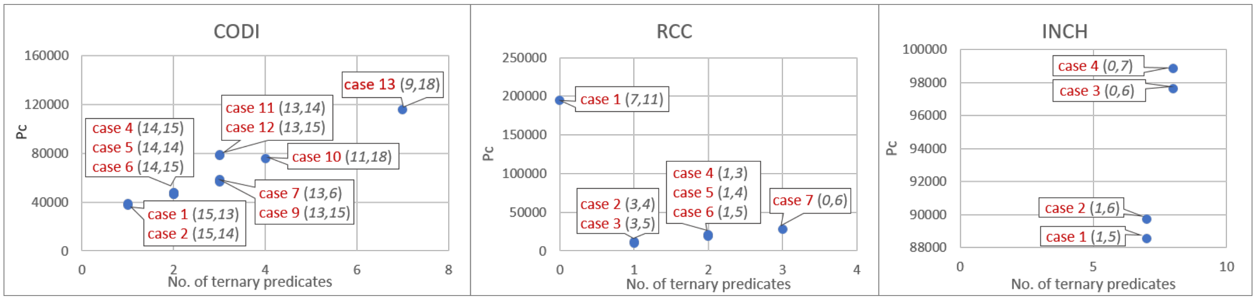


Figure 6.7: Graph showing the variation of P_c for each case with increasing size of terminology in an ontology with $d = 20$, $r = 5$ for CODI, RCC, and INCH. The x-axis is the number of ternary predicates, the y-axis P_c and the cases shown as dots along with the number of unary and binary predicates included in them. The number of predicates are from the clausification of the TBox and basic ABox (i.e. for $r = 1$).

6.3 Guiding Predicate Selection for ODE

In order to reduce the number of propositional variables that determines the search space, we can try to reduce the signature of the ontology, including the number of additional predicates added from clausification, and to reduce the number of propositional clause that determines how quickly the solver terminates, we can try to reduce the number of FOL variables per clause in the FOL-CNF formula, and reduce the overall number of FOL-CNF clauses.

The structuring of predicates in a problem is very ontology dependent. Definition elimination depends on the dependency between defined predicates, with elimination starting from the predicates on which no other predicates depend (i.e. at the lowest level in the graph) and then moving up. If a predicate is not ideal for elimination, which is decided based on size measures of the FOL-CNF formula, the pointer skips this but can move up to the next connected predicate. Unlike typical formula simplification techniques, ODE may not always reduce the size of the problem. There are two ways of potential growth in size of the SAT representation: (1) larger P_v through skolemization: see the sum of the P_v from the TBox and ABox in case 1 for RCC in Table 6.1.2, (2) larger number of FOL-CNF clauses, (3) FOL-CNF clauses with high variable-density - $v > 2$, (4) FOL-CNF clauses with width ≥ 3. While ODE typically reduces the number of propositional variables in the SAT problem in a way that improves solver tractability, there are two things to be careful about with the growth in propositional clause set. Firstly, a large propositional clause set impedes scalability significantly – it carries a potential for a significant slowdown, because each clause takes up valuable memory and needs to be looked at during the propagation phase after each variable assignment. Secondly, a clause becomes vital in a search process only when it becomes unit, but longer clauses are more difficult to become unit. ODE should not be applied when it results in a significant increase in longer clauses, wider clauses or clauses with more variables. All of this can be easily measured on the FOL-CNF versions of the different ontologies, which can help select the best set of definitions to eliminate such that the simplification is maximally efficient.

6.4 Discussion and Conclusions

Towards addressing objective 3 of this dissertation (O3 in Section 1.2.2) we have studied the growth of the size of an ontology's SAT translation in terms of the number of propositional variables and clauses with respect to the size of the signature or ontology vocabulary (after conversion to clausal form), the model domain size, and the number of relational assertions in the ABox. The study verified the hypothesis that aggressive ODE on predicates of highest arity mostly yields a significant reduction in the number of propositional variables and a reasonable reduction in propositional clauses, but sometimes depending on the definition being eliminated, ODE may be detrimental. For example, the definition of the optionally defined term *NTPP* in RCC is defined using three other optional definitions: *EC*, *O*, and *P*. The elimination of *NTPP* (including the 3 other dependent predicates) results in the nesting of terms in the corresponding definiens and defined assertions. Such sentences with deep terms either lead to an FOL-CNF formula with higher number of clauses, variable-density or even formula-width. Transformation simplification (used in Paradox and Mace4) adds new function symbols that replace deep sub-terms inflating the number of predicates even more, or increases FOL-CNF clause count from clause splitting rules to transform long clauses with many variables into several flat clauses with fewer variables. Thus, the most eager definition elimination that is theoretically possible will likely not be the best choice, and this motivates the next chapter, which analyses the model finding performance for these different cases and compares it to the calculated measures. During ODE it is also important to be aware of the number of predicates present in the DBox but also be cognizant of any additional predicates that may be introduced in the ABox from skolemization. Existential quantifiers (in the definiens and defined assertions) play a huge role as they create new predicates after skolemization – but as consequence, we can use the FOL-CNF ontology (the translation to FOL-CNF being polynomial in time, not exponential) to fairly cheaply measure this and pick the best set of eliminated definitions *before* starting the time-intensive model finding task. On that note, ODE is efficient only when the number of auxiliary predicates included is minor compared to the number of optional definitions being eliminated.

CHAPTER 7

EXPERIMENTAL STUDY OF THE EFFECT OF ODE ON MODEL FINDING TIMES

In order to understand the correlation between the theoretical measures of the size of SAT problems from FOL ontologies that we formalized in Chapter 5 and the hardness of real-world problems in practice, we conduct model finding experiments to verify the external consistency of FOL ontologies (specifically spatial ones) with (spatial) datasets, through applying ODE with different levels of aggressiveness. In Chapter 6 we designed TBoxes (or cases) for three spatial ontologies – CODI, RCC and INCH – that only differ in which definitions are included or removed to study the tradeoff in reduction in P_v from the TBox and potential increases in P_v (and P_c) in the ABoxes. In this chapter, we construct multiple versions of ontologies for these TBoxes using real-world datasets. The different cases for an ontology for a specific dataset generates models that do not semantically differ, as the extensions of the defined predicates are unique and can be reconstructed. Our experiments are specifically designed to test the following hypothesis "*optional definitions in the TBox significantly impact FOL model finding time, and therefore eliminating them and rewriting ABox facts that use them with their definiens allows improved performance.*".

Towards objectives 3 and 4 (O3, O4 in Section 1.2.2) of this dissertation, we conduct an empirical investigation with these ontology instances to study the effectiveness of ODE as reflected in the run-times of three model finders: Paradox, Vampire and iProver, which have consistently been either the winner or the top contenders in the relevant divisions of the CASC ATP competitions [219, 265] (see details in Chapter 3). Moreover, the idea behind the design of experiments is to also systematically study how the growth of ABox size by regulating the number of individuals (d) and relational assertions (r) impacts model finding time. Through systematic study we demonstrate the linkage between the calculated size measures – studied in detail in Chapter 6 – and practical model finding performance, and through correlation analysis validate our findings. The results presented in this chapter is an important step towards the more general goal of improving the feasibility and scalability of practical SAT-based FOL model finding.

7.1 Design of Study

In this section we explain the design of the study, whereby we construct ontologies using different definition sets (from the DBoxes of each ontology) and different sized datasets to validate the hypothesis stated above. These sample ontologies also serve as important practical benchmarks (that is, instances generated from real-world datasets in the spatial domain) in the evaluation of automated theorem provers. For each TBox (cf. Tables 6.1.2, 6.1.2, and **??** - 13 cases for CODI, 7 for RCC, and 4 for INCH) we use a Python script[1] to construct sample ABoxes of different sizes[2] as described below.

7.1.1 Constructing (r-d) ABoxes

ABoxes with controlled r values don't come naturally but are crucial for a good comparison of problem size. Thus, we have to artificially create them using a stratified sampling technique. For each combination of d and r, 10 sample ABoxes are constructed from a single *master dataset* about the critical habitat for lynx in Maine[3] . Figure 7.1 shows the map from which geometric entities and relations between them are extracted. Detailed spatial information within this extent is abstracted from GIS shapefiles from the Maine Office of GIS Data Catalog[4] : points represent schools and endpoints of road segments, lines represent road segments, and regions represent the boundaries of towns, subdivisions and counties. Some sample assertions are as follows:

[1] `https://github.com/shirlystephen/SpatialModelFinding/PythonScripts`
[2] We are only interested in computing finite models having a finite domain.
[3] Unit 1 from `https://www.gpo.gov/fdsys/pkg/FR-2014-09-12/pdf/2014-21013.pdf`
[4] `https://www.maine.gov/megis/catalog/`

- sf_point(‘*FoxcroftAcademy*’)
- sf_line(‘*road_I95*’)
- sf_region(‘*PiscataquisCounty*’)
- intersects(‘*FoxcroftAcademy*’ ‘*PiscataquisCounty*’)
- within(‘*segment1103*’ ‘*road_I95*’)
- crosses(‘*segment1103*’ ‘*PiscataquisCounty*’)

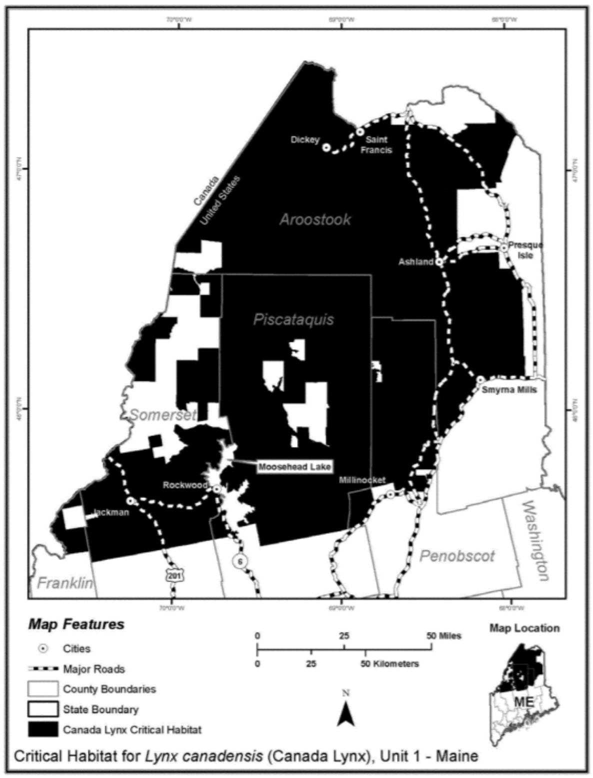


Figure 7.1: Geometric map about the critical habitat for lynx in Maine from which the master dataset is constructed.

The master test suite describes the spatial relationships between 425 spatial objects (i.e. *individuals*) using 130,256 ground assertions (4,937 positive ones and 125,319 negated ones), each of which uses a single unary predicate (*Point*, *Curve*, *ArealRegion*) or single binary predicate (*within*, *overlaps*, *intersects*, *crosses* and *touches*) from the Simple Features (SF) standard as axiomatized in FOL as SFA-FOL in Chapter 4. A statistical summary of the ABox (number of positive/negative facts from each concept and relation) is provided in Table 7.1. During construction of the sample ontologies, these SFA-FOL terms are replaced with the respective terms from CODI, RCC, and INCH (see mapping between terms in Table A.1 in

	Unary-Concept Assertions (425)			**Binary-Relational Assertions** (130,256)				
	Point	Curve	ArealRegion	within	crosses	overlaps	intersects	touches
positive	194	42	189	227	414	1947	1038	1311
negated	0	0	0	60947	1160	30152	30222	2838

Table 7.1: Content of the master ABox from which sample (r-d)ABoxes are constructed for CODI, RCC and INCH.

the appendix). Then distinctness assertions are added to the constructed ontologies to ensure that all selected individuals are actually distinct. The complete set of sample ontologies constructed in this study is available as CLIF files in the github repository[5] .

The master ABox contains surplus assertions for each optional predicate needed for the desired study of problems in our study so we don't run out of assertions during the sampling process.

Stratified Sampling Process

We construct (r-d)ABoxes for each case in a theory using a stratified sampling approach as follows:

1. An assertion for a binary predicate in DBox D is selected at random and the two individuals participating in the relation are added to a list M. Then we pick an assertion for all other optional predicates in D (individuals in each selected assertion are added to M in succession), such that each assertion contains atleast one element from M and one positive and one negative assertion for each optional predicate in D have been added to (r-d)A.

2. Step (1) is repeated until $|M| = d$ or the number of assertions selected for each predicate reaches r. If the ABox has realized the size d first, random assertions for each optional predicate are chosen from the master ABox containing individuals in M and added to (r-d)ABox until the desired number of r assertions for each predicate is achieved. Otherwise,

[5] `https://github.com/shirlystephen/SpatialModelFinding/SampleDatasets`

if the ABox has realized the size r first, two randomly selected individuals are removed from M and all assertions containing these individuals are dropped from (r-d)ABox and step (2) is repeated until (r-d)A reaches its desired size.

Note that the sample ABoxes are stratified in the sense that all non-unary optional predicates are used equally. While this may rarely happen in practice[6] , it allows us to rule out many other factors in our analysis of the growth of the resulting SAT problem as well as experimental model finding times.

7.1.2 Constructing Defined (r-d) ABoxes

ODE is applied to each sample (r-d)ABox to rewrite sentences that use predicates that are already removed from TBox that it uses. By the ODE rule, sentences that uses eliminated definitions are replaced by their defined assertions (cf. Example **??**), provided the substitution is made throughout wherever the predicate appears in the ABox, resulting in a non-factual ABox with ground and/or partially-ground first-order assertions. For example, the default TBox (case 13) of CODI includes the following explicit FOL definition for *Inc*,

$$Inc(x,y) \leftrightarrow \exists z[(Cont(z,x) \wedge P(z,y) \wedge z <_{dim} x) \vee (P(z,x) \wedge Cont(z,y) \wedge z \prec_{\mathrm{dim}} y)]$$

This defined predicate *Inc* is not used in any other axioms and definitions and thus can simply be removed from the TBox to reduce its set of binary predicates (the CODI cases 1-7,11,12 all remove *Inc*). Now any assertion in a (r-d)ABox that uses *Inc* must also be rewritten for the CODI cases 1-7,11,12. For example, the assertion $Inc(`exit193', `i95')$ will be rewritten as: $\exists z[(Cont(z, `exit193') \wedge P(z, `i95') \wedge z <_{dim} `exit193') \vee (P(z, `exit193') \wedge Cont(z, `i95') \wedge z <_{dim} `i95')]$.

[6]To estimate the size of SAT problems resulting from practical datasets, we would need to treat r as an upper bound on the number of assertions for any individual predicate. But as it turns out, r primarily influences the number of propositional clauses but rarely the number of propositional variables.

7.1.3 Experimental Environment

We used the latest versions of three state-of-the art model finders – Paradox[7] [164], Vampire[8] [172] and iProver[9] [165] – for our work. For all three solvers we used the default model finding option, which is the `casc_sat` mode for Vampire and the `sat` mode for iProver. We used a timeout of 50,000s, 20,000s, 20,000s for Paradox, Vampire and iProver respectively. All experiments are conducted on an Intel Xeon CPU E5-2620 v3 at 2.40 GHz (with 12 cores, though a single instance of any solver does not use more than a single core) with 64GB RAM and 64bit Windows 10 Pro, using Ubuntu (release 16.04) inside an Oracle VirtualBox VM (version 6.0) with 40 GB of allocated RAM and 12 CPU processors.

7.1.4 Statistical Analysis Methods

Each model finder was run with ten different samples for each case of each ontology and each combination of a domain size (ranging from 10 to 50) and an r value (5, 10, 15, or 20; INCH samples include 8, 12, 18) for a total of 2,080, 840, and 560 problems of different sizes for CODI, RCC and INCH, respectively. In each sample set sometimes there aere a number of outliers, which took disproportionately longer[10] . For example, for case 7 in CODI, when $d = 30$ and $r = 15$, the runtime of Paradox for only two of ten samples is over 1,500s, while the remaining samples have runtime ranging between 180s and 900s. Therefore we only plot the *low-mean* of each sample set of ten samples calculated as follows:

$$S = \text{Set of (tractable) model finding times for a case and its (r-d) ABoxes,}$$

$$S_L = \{s_i \in S | s_i < (\mu + \sigma)\}; \text{ where } \mu = \frac{\sum_{i=1}^{n=|S|} s_i \in S}{|S|} \text{ and } \sigma = \sqrt{\frac{\sum_{i=1}^{n=|S|} ((s_i \in S) - \mu)^2}{|S|}}$$

$$\text{then, } \textit{low-mean} = \frac{\sum_{i=1}^{n=|S_L|} s_i \in S_L}{n}$$

[7] accessed on 02/10/2018 - `https://github.com/c-cube/paradox`

[8] accessed on 01/12/2020 - `https://github.com/vprover/vampire`

[9] accessed on 01/12/2020 - `http://www.cs.man.ac.uk/~korovink/iprover`

[10] The stratified sampling technique for creating these samples does not allow us to control for the hardness of the samples, some end up significantly harder than others

The low-mean is the average runtime of all samples that terminated within less than the mean μ of all ten samples plus one standard deviation ($\mu + \sigma$) runtime for that sample set. This time is representative of how long it takes for verifying the ontology (specifically the theories here) against the majority of samples. Cases where the majority of problems did not terminate are assigned the solver timeout and are specially marked in our graphs (the percentage of intractable, i.e. non-terminating, samples for each case in the three ontologies is presented in Table **??** in the appendix). For some cases in Paradox and Vampire, where the majority of problems terminated but without generating a model due to a memory error (likely due to reaching some internal memory limit), we use the solver runtime, though these times do not significantly effect the overall trend.

7.2 Experimental Results

In this section we present the model finding times and discuss any trends with respect to the findings from Chapter 6. Figures 7.5, 7.3 and 7.6 present the runtimes for the three model finders for the different cases in CODI (13 cases), RCC (7 cases) and INCH (4 cases) for different (r-d)ABoxes, where each line in a single plot represents the low-mean runtimes for a specific r. We will discuss specific observations and trends and how they relate to the P_v and P_c values. We first analyze the results for Paradox and Vampire in more detail for each of the three ontologies, as they render similar trends. Afterwards, we look at iProver as its results are very different. Finally we presents statistical correlation results between the empirical findings and theoretical measures.

7.2.1 Paradox and Vampire Results

CODI: The results from both Paradox and Vampire show that runtime seems to exponentially increase with d. More interestingly for an (r-d)-ontology, runtimes also significantly increases in cases that include more definitions, as predicted by their increases in P_v and P_c. This is especially obvious for Paradox, where for the default case – case 13, which includes all five optional definitions (for a total of 13 binary predicates), the number of propositional variables is 63,380. This is over four times the number of propositional variables from case 1 (13,980), and the runtime for case 13 more than quadruples compared to case 1: e.g. for domain size 20, runtime increases from 7s to 345s and from 134s to 713s for $r = 5$ and 20, respectively. The exponential increase in runtime is more obvious, for domain size 30, the runtimes increase from 16s to 32,000s and 164s to over 50,000s, which is the timeout at which point the model finder is told to terminate. While Vampire is consistently faster than Paradox, the model finding times of both exhibit a very similar pattern that is also closely correlated with the number of propositional variables as visualized in Fig. 7.5. The reduction in the model finding time between the default case and the best case can be dramatic in this ontology: Vampire shows upto 27 times runtime increase for some (r-d)-problems (cf. Fig. 7.7), while the decrease for Paradox is even higher - an decrease in three orders of magnitudes. This also becomes evident from the size of models that can be constructed (cf. Fig. 7.5): in the default case, models of size 30 and 50 are the limit for Paradox and Vampire, respectively, whereas the best case allows constructing models of sizes up to 120 individuals in similar times as previously needed for size 30 (cf. Table 7.2 for model finding time using Paradox for case 1 in CODI for domain sizes 100 to 120, $r = 5$). While there are slight differences about how well certain cases perform (e.g. cases 11 and 12 are more difficult for Paradox, whereas cases 8 to 10 are more difficult for Vampire), invariably the default case consistently takes the longest to construct a model for both solvers and quickly becomes intractable from $d = 30$ (for Paradox) and $d = 50$ (for Vampire) on.

Domain size	100	110	120
Time in s	8,564	9,434	25,704

Table 7.2: Model finding time using Paradox for case 1 in CODI for d 90 to 120 ($r = 5$).

Overall, case 1, which removes the most definitions i.e. performs ODE most aggressively, yields the best runtimes for Paradox throughout. However, the results for Vampire show that removing as many definitions as possible does not always result in the best performance, in fact, case 2 that retains the definition of PP performs best. Another critical factor is the complexity of a definition. For example, cases 2, 3, 5, 8, and 11 all include exactly one additional definition (PP, C, PO, Inc, and SC respectively) compared to case 1, but lead to different speed-ups. Vampire's and Paradox's runtimes increase more when adding Inc or SC as compared to when adding C or PO, which are simpler because they only contain one existentially quantified conjunction each. Whereas Inc contains a disjunction of two existentially quantified statements, and SC contains a conjunction of one existentially quantified and one universally quantified statement (cf. Section 2.4.2.1 for their axiomatization). In fact, removing only the predicate Inc and its definition yields a 88/59% (Vampire/Paradox) and only SC a 79/75% decrease in runtime for $d = 40$ and $r = 10$. This holds similarly for other d and r combinations, and in fact for larger values, problems containing these definitions are the first that become intractable. One explanation is that Inc or SC add additional FOL-CNF clauses in the TBox, which, in the case of Inc are rather wide (i.e. with more than 3 literals, C_T with $w \geq 3 = 43$, cf. column 10 in Table 6.1.2) and, in the case of SC have high variable-density (i.e. contain more than 3 variables, C_T with $(v = 3) = 5$, column 6 in Table 6.1.2). And both definitions are not used in any other definitions, which would potentially reintroduce additional Skolem functions. Such complexity measures could potentially be used to decide which defined predicates are prime candidates for removal but require additional experimentation beyond the scope of this work.

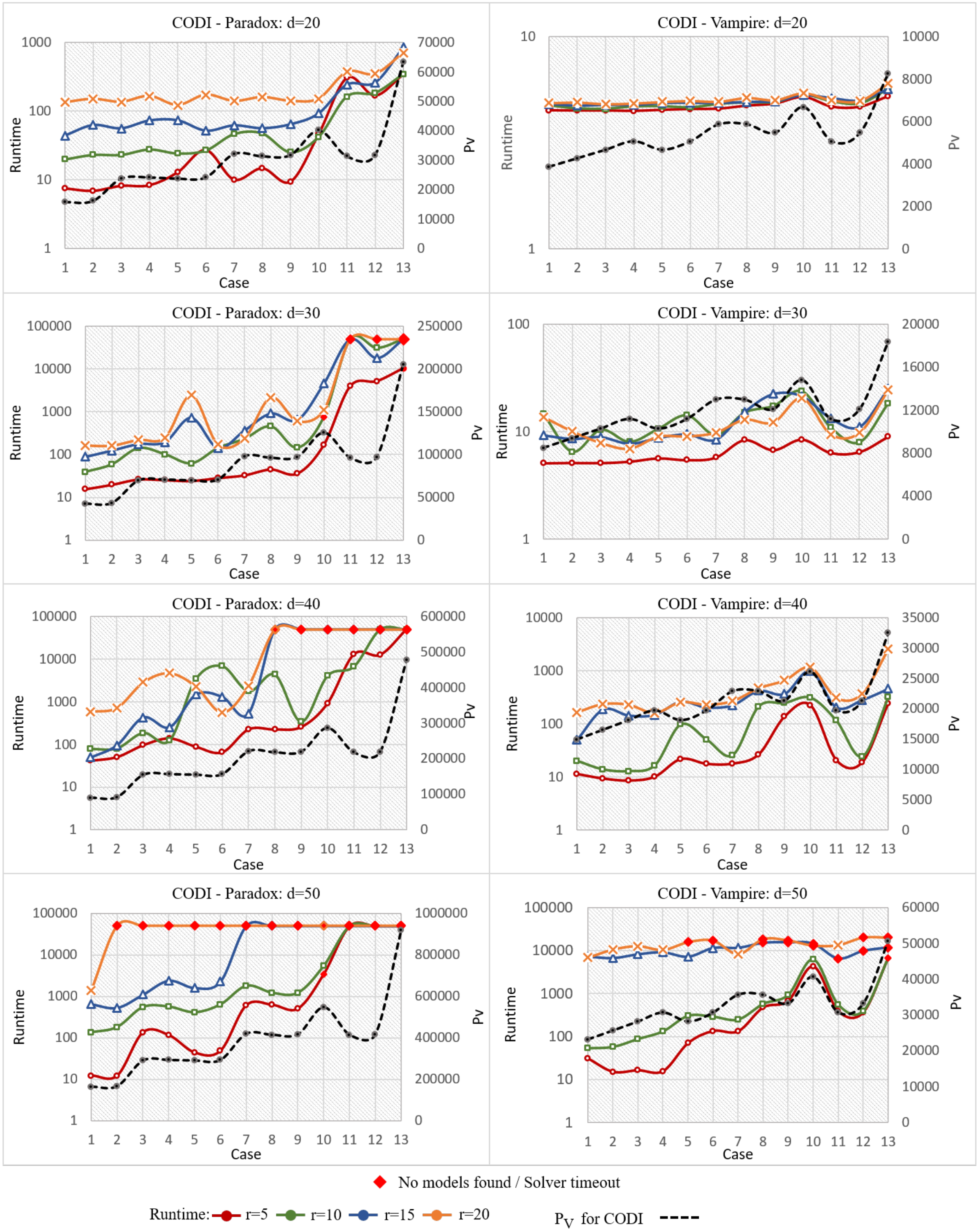


Figure 7.2: Model finding times for CODI (domain sizes 20 to 50) using Paradox and Vampire(cf. Tables **??** and **??**). Each graph fixes the domain size and each line represents an r value. Cases are on the x-axis, from case 1 with a total of 8 (binary) predicates to the default case (case 13) having a total of 13 binary predicates. The runtime (y-axis) uses a log_{10} scale, and P_v in the secondary axis uses regular scale. Note: P_v for CODI does not change with different r values.

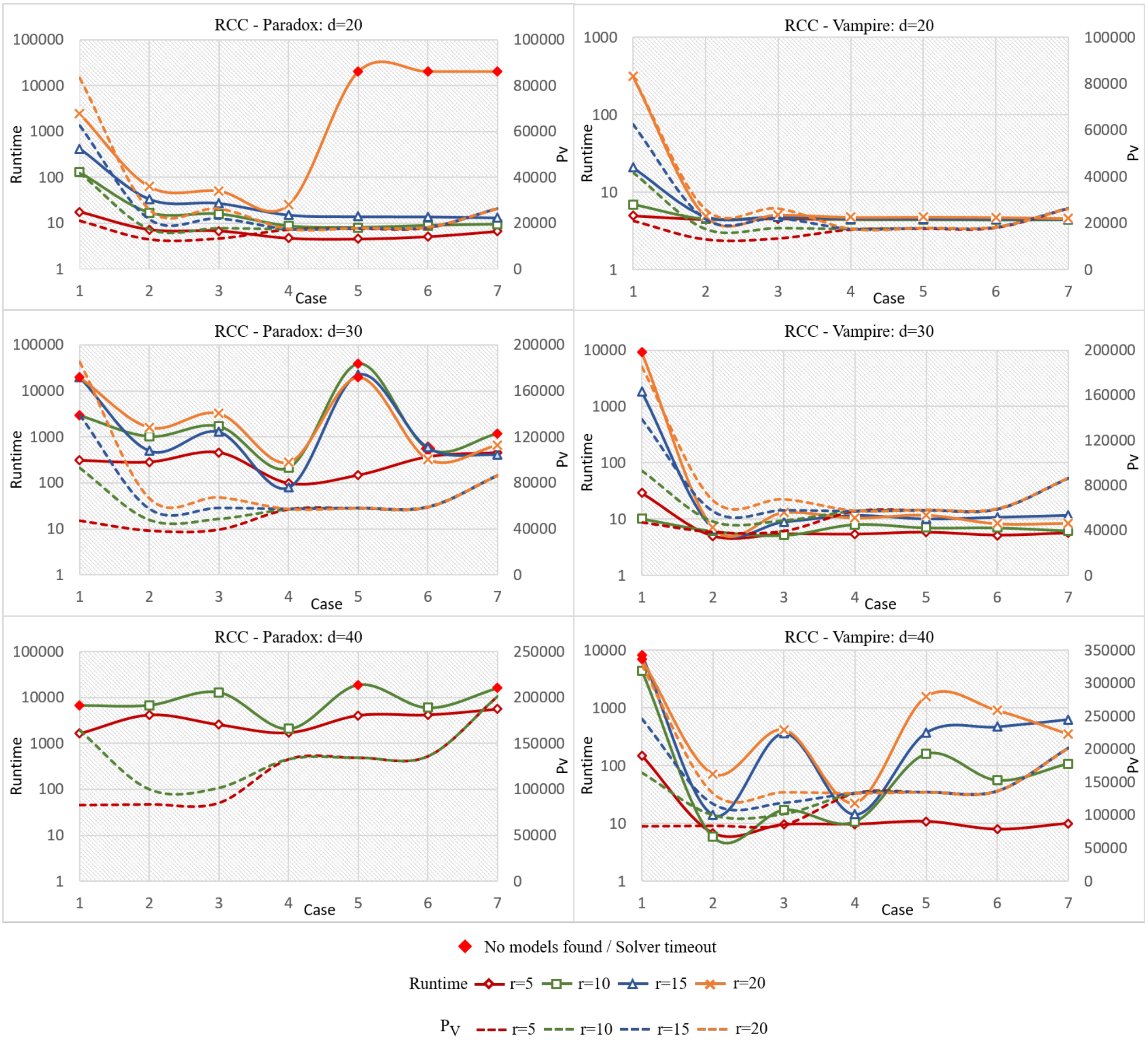


Figure 7.3: Model finding times for different d and r values for the different cases of RCC (cf. Table **??** in the appendix). The cases along the x-axis are sorted by increasing number of defined predicates. The runtime (y-axis) uses a log_{10} scale, and P_v in the secondary axis uses regular scale. Runtimes from iProver for RCC are not displayed as it did not find any models at all.

RCC: For RCC, a slightly more nuanced story emerges. While the runtimes mostly follow the trend of P_v, the steep increase in P_c and P_v in case 1 (cf. Fig. 6.3) yields comparable and sometimes even worse runtime on some (r-d)-problems than performing no ODE at all.

As predicted by the theoretical analysis, the steep increase in P_v and P_c when removing all definitions (case 1) makes it the most difficult case, besides the default case, for both solvers. RCC is an excellent example of the impact of clauses with $w \geq 3$ on model finding – as seen by the visual correlation (cf. Fig 7.3, statistical correlation results are discussed in more detail in the next section) between case 1 having more clauses with high formula-width (C_A with $w \geq 3 = 32$, cf. column 15-18 in Table 6.1.2) on runtimes. P_v and P_c are the lowest in case 2, which removes all optionally defined predicates except for P, but keeps both the number of newly introduced Skolem functions and the number of clauses with more variables relatively low. This is the best case for Vampire for both domain sizes 20 and 30. Paradox performs slightly better on case 4, which additionally retains O and results in even fewer clauses with more variables ($C_{A,2}$ is 1 compared to 8 for cases 2 and 3). As the number of defined predicates further increases to 4 and beyond (cases 5–7), the runtime increases again. This phenomenon is similarly observable for Vampire, especially for $d = 40$, which is also the domain size beyond which conspicuous differences in runtime for the different cases is visible. Similar to CODI, with the best case, although Vampire runs longer, it scales better compared to Paradox with the capability to find models on larger (r-d)-problems.

INCH: (Note: The study with INCH was not the emphasis of our work, but added as yet another ontology for comparison to see whether some of the trends from CODI and RCC transfer to this ontology.) Even though INCH includes only few definitions, its clausification yields an extremely large number of FOL-CNF clauses and additional predicates (from Skolem functions) with high arity ($a \geq 2$), which eventually results in large P_v and P_c even for small domain values. For example, when $d = 20$ and $r = 5$, even with the most aggressive ODE, i.e. for case 1, $P_v = 58{,}100$, which is 4 and 3 times the smallest values of P_v for CODI and RCC respectively for the same domain size and r value. We therefore had to experiment with smaller domain sizes ($d = 10$ and $d = 15$) to obtain any models at all. Overall, the runtime of Paradox is lowest for cases 3 and 4. When $d = 10$, Paradox's performance is mostly uniform across cases (but the runtimes are also too short to make any meaningful distinctions), and when $d = 15$ the removal of certain definitions, particularly CO, deteriorates its performance. With Vampire, the improvement in runtime with ODE is significant with larger problems (i.e. from $d = 15$ and $r = 8$ onwards), where the default case has upto 4-times higher model finding time compared to case 1. In addition the variation in Vampire's runtime for the default case (case 4) across different r values mimics the phase transition of random SAT. Model finding time is less when the problems are less constrained ($r = 5$) or heavily constrained ($r = 20$), compared to when $r = 10, 12$ or 15. For example, when $d = 15$ and $r = 15$, the runtimes for case 4 and case 1 are 1523s and 399s respectively, but when $r = 20$, the runtimes for the two cases decreases to 310s and 240s respectively.

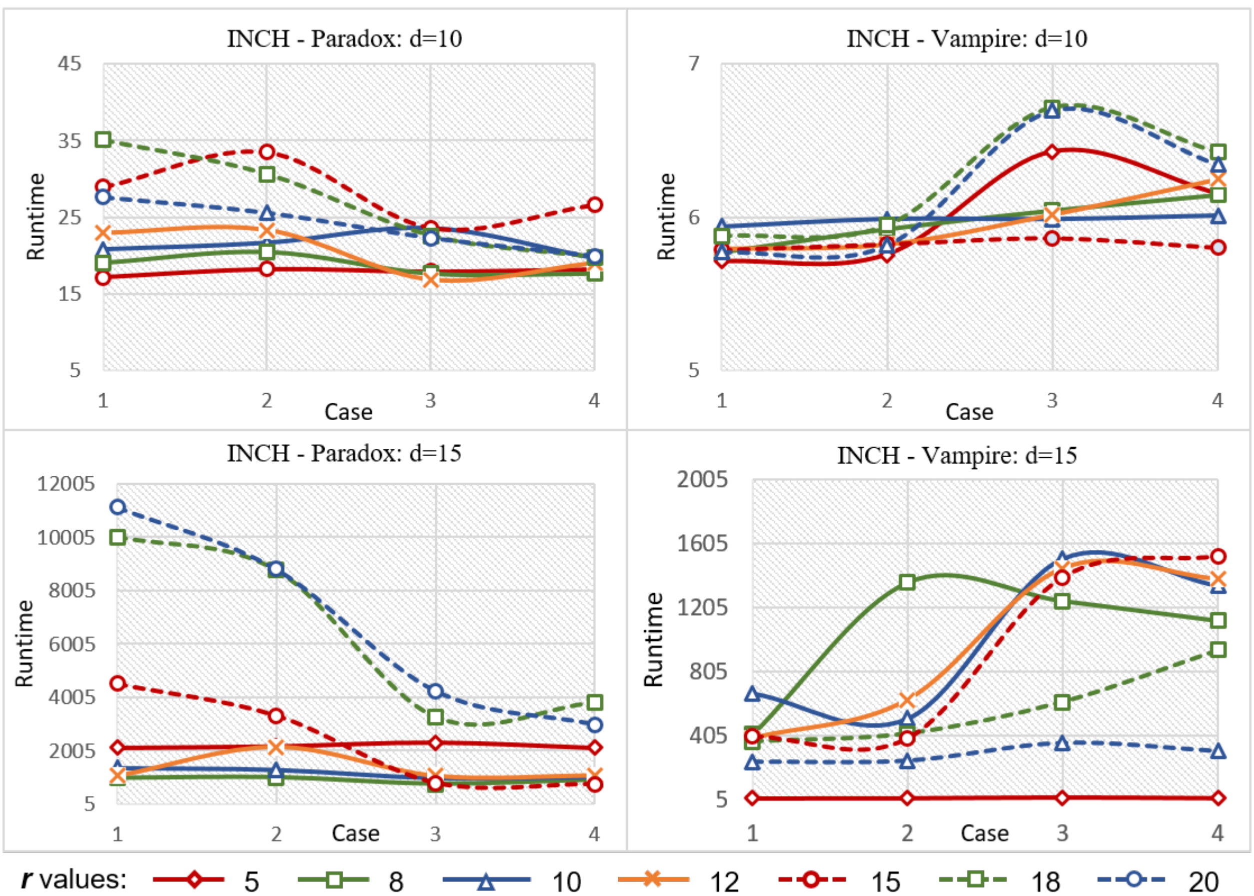


Figure 7.4: Model finding times for different d and r values for the different cases of INCH using Paradox and Vampire (cf. Table ?? in the appendix). The cases along the x-axis are sorted by increasing number of defined predicates. Runtime (y-axis) is in regular scale.

7.2.2 IProver Results

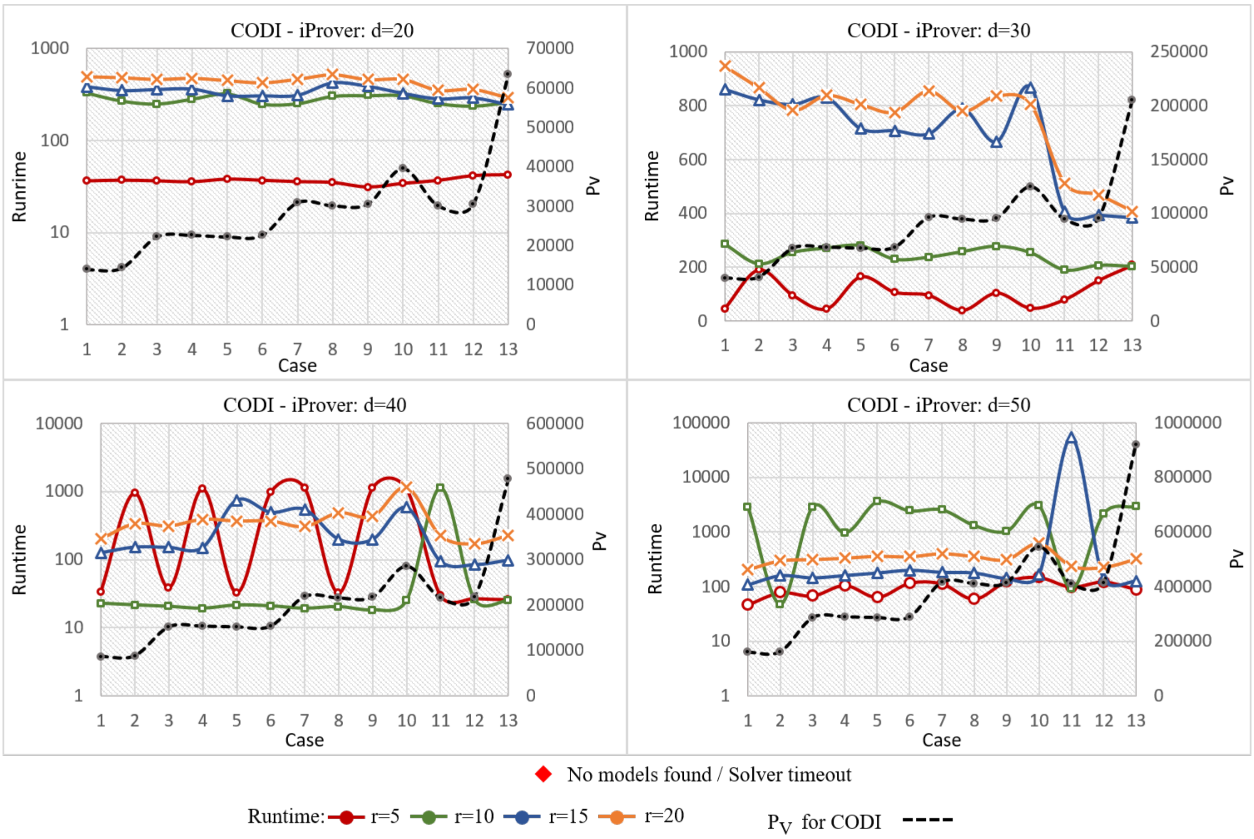


Figure 7.5: Model finding times for different problems for CODI (domain sizes 20-50) using iProver (cf. Table ??). Each graph fixes the domain size and each line represents an r value. The cases are listed on the x-axis, from case 1 with only a total of 8 (binary) predicates to the default case (case 13) with 5 additional defined predicates a total of 13 binary predicates. The runtime (y-axis) uses a log_{10} scale, and P_v in the secondary axis uses regular scale. Note: P_v for CODI does not change with different r values.

IProver exhibits much less predictable results across the different ontologies, cases and problem sizes. For CODI, iProver overall performs much better than Paradox and Vampire with the exception that Vampire's best case performs better for $d = 20$ to 40. Unlike Paradox and Vampire, the default case is not the worst case, and case 1 is not always the best case. In fact, in most problems the model finding times for these two cases are relatively close. Thus, for CODI it seems that iProvers built-in predicate elimination (cf. Section 3.2.3) performs well. However, very different results emerge for RCC and INCH (cf. Fig. 7.3): on RCC, iProver fails to produce any models whereas on INCH it performs much worse than Paradox and Vampire for $d = 10$ and 15 and it altogether fails to produce models for $d = 15$ at r values of 18 and 20.

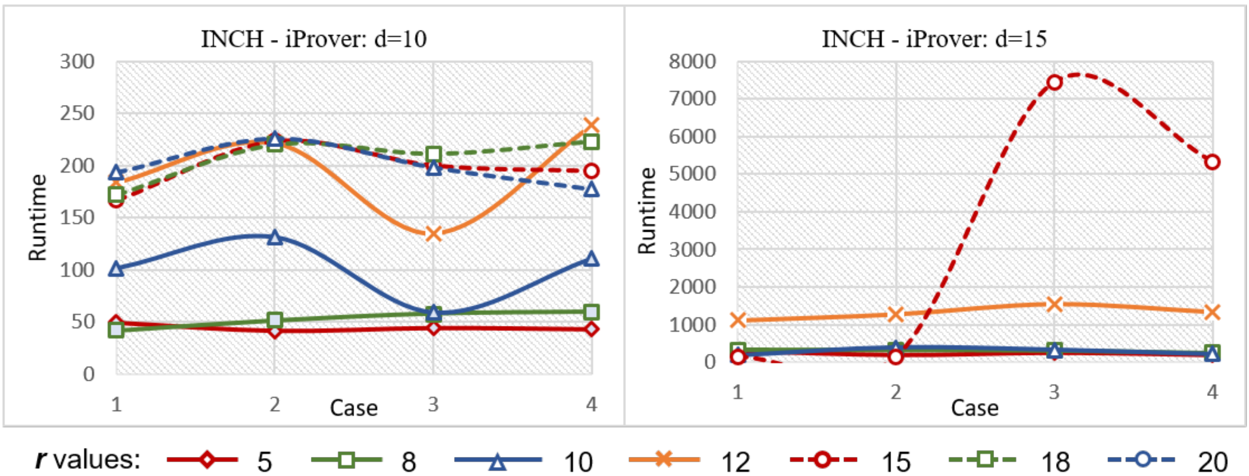


Figure 7.6: Model finding times for different d and r values for the different cases of INCH using iProver (cf. Table ?? in the appendix). The cases along the x-axis are sorted by increasing number of defined predicates. Runtime (y-axis) is in regular scale.

7.3 Analysis

Now we try to further strengthen our hypothesis by determining that there exists an exponential relationship between practical model finding time and theoretical measures of an ontology's size, and also reveal that significant gains in runtime can be achieved through definition elimination.

7.3.1 Correlation Analysis between SAT Problem Size and Model Finding Times

In Chapter 6 we showed that the number propositional variables and clauses in an ontology $\mathcal{O}$ with a (r-d)ABox increases significantly with the signature of $\mathcal{O}$, specifically the number of binary defined predicates, and d. Through empirical analysis we demonstrated that model finding time looks exponential with respect to P_v. Here, through correlation analysis we statistically verify how the practical *hardness* – measured in terms of model finding times – of ontologies with a (r-d)ABoxes corresponds to the size of their SAT translations. Specifically we calculate the correlation between three transformations (linear, logarithmic - log_{10}, and square root) of the low-mean model finding time over all cases and all (r-d)ABoxes against three theoretical measures of their size: P_v, P_c, and P_w (the approximate percentage of P_c with width ≥ 3 calculated using the following formula – $\left(\frac{C_{T,w\geq3}}{C_T} + \frac{C_{A,w\geq3}}{C_A}\right) \cdot P_c$[11] . Correlation values are calculated only using results from tractable problems for all ontologies and provided in Table 7.3. The highlighted values (cells in gray) are the runtime transformations that have the most significant correlations with size. Like expected, the results prove that there is an exponential relation between model finding time with P_v, P_c, and P_w (except iProver's results for CODI, which show a more linear relation for P_v, and a square root relation with P_c and P_w). The complete scatter plots of runtime and the three measures of size are presented in Figures A.1, A.2, and A.3 in the appendix.

Both Paradox and Vampire show a positive correlation between runtime and all three measures of size, somewhat higher than 0.5 for CODI. While Vampire shows a strong positive correlation with size for CODI and INCH, its values for RCC are somewhat low. In RCC although we predicted that larger P_c and higher width of clauses (as occurs in case 1) have a more signficant influence on runtime, the correlation results for the two measures are comparatively lower than for P_v, with both Paradox and Vampire. With large (r-d)ABoxes, for RCC's case 1, both Paradox and Vampire are completely intractable, and therefore measures for this case is largely not unaccounted for while determining correlation. Although the iProver's time is positively correlated with the size of CODI's problems, the correlation values of runtime with P_c and P_w are too low to indicate an exponential relationship. However

[11]In this chapter and the previous we occasionally compared the size of problems in terms of their formula-width with the hardness of model finding to highlight some difficult cases. But we leave the detailed examination of the impact of W on the size of the SAT problem for future work.

Ontology	Prover	Linear			Exponential			Quadratic		
		P_v	P_c	P_w	P_v	P_c	P_w	P_v	P_c	P_w
CODI	Paradox	0.19	0.24	0.17	0.52	0.53	0.48	0.38	0.42	0.35
	Vampire	0.37	0.37	0.42	0.78	0.79	0.82	0.53	0.54	0.59
	iProver	0.89	0.12	0.15	0.78	0.08	0.08	0.15	0.18	0.70
RCC	Paradox	0.43	0.06	0.03	0.82	0.32	0.27	0.70	0.19	0.16
	Vampire	-0.07	-0.04	-0.01	0.40	0.31	0.31	0.01	0.06	0.08
INCH	Paradox	0.54	0.55	0.64	0.95	0.95	0.97	0.78	0.79	0.84
	Vampire	0.73	0.72	0.59	0.91	0.91	0.86	0.83	0.82	0.73
	iProver	0.40	0.39	0.29	0.63	0.63	0.56	0.52	0.51	0.42

Table 7.3: Correlation analysis results between runtime of three model finders and three measures of the size of SAT problems for CODI, RCC, and INCH: no. of propositional variables, no. of propositional clauses and approximate number of those clauses having three or more literals.

we also did not expect to observe any exponential growth of runtime with the size of the problem using iProver, because of its fluctuating behaviour across the three spatial ontologies, while its performance was superior to Paradox and Vampire for CODI (smaller runtimes and scaled efficiently with larger domain sizes), it was altogether intractable on any of the tested problems in RCC, and did not scale as successfully as Paradox and Vampire for INCH. Surprisingly, although it was hard to recognize a uniform meaningful decrease in runtime with eliminating definitions in INCH, the strong correlation results (with all three solvers) prove that ODE actually improves solver performance. Interestingly we also found that unlike for regular CNF problem [278], problems generated from any of the three ontologies did not reveal any relevant correlation between clause-density ratio (P_c/P_v) and any solver performance – see results in Table **??** in the appendix[12] .

[12]We have reviewed in related work in Section 3.2.1 that many existing works in propositional logic have found a strong strong correlation between the hardness of the problem and clause-density ratio, but we do not observe a replication of this with FOL ontologies.

7.3.2 Speedup in Model Finding through ODE

The idea here is to analyze runtime improvements of the best case (i.e. the case with the lowest relative runtime over all r and d values for all ontologies, the second worst case (to determine the mimimum improvement from eliminating definitions), and the average improvement over all cases using ODE from the default case. To reiterate, case 1 in a theory corresponds to the case that removes all optional definitions from its DBox, whereas the default case includes all optional definitions. Usually case 1 reduces P_v and P_c the most (except RCC, where P_c for case 1 is larger compared to the default case) and is therefore theoretically the best case. However, in practice the best case may vary for different solvers and therefore the preprocessing algorithm for each solver must be optimized to select the ideal set of definitions for elimination. Figure 7.7 shows the maximum, minimum, and average runtime decreases from eliminating definitions for the three ontologies.

CODI: Paradox performs the best on case 1 with a runtime improvement that approaches 100% especially for larger domain sizes, and an average runtime of always more than 50%. Although Vampire performs well on case 1, the relative decrease in runtime is higher for case 2, i.e. with the inclusion of the definition for PP. With case 2, Vampire achieves a maximum runtime improvement close to 98%, and an average runtime improvement as high as 73%. The runtime improvement of iProver with case 1 is mostly negligible, and the average improvement is also rather low[13] .

Since there does not seem to be a best case, we decide that ODE is not ideal for CODI with iProver.

RCC:[14] Overall, Paradox performs best on case 4, which is expected, as it has fewer binary and unary predicates in the FOL-CNF formula (aggregated predicates from the ontology and from skolemization) compared to case 2 and 3 (cf. Table 6.1.2), while Vampire performs best on case 2, which has the smallest ontology signature. The runtime gains with

[13]There also isn't an average increase in runtime.

[14]We remind the reader that we only have results from Paradox and Vampire for RCC, since iProver was altogether intractable on the tested ontologies.

the best cases using both solvers is significant with larger domain sizes indicating that ODE is capable of pushing the limits of scalable model finding.

INCH: ODE shows no improved runtimes with Paradox, however this cannot be generalized, since we are unsure if with better hardware capabilities ODE might reflect a different trend with larger (r-d)-ontologies[15] . iProver performs considerably well with case 1, which also happens to be its best case. Although it is hard to see a substantial improvement in runtime with Vampire for lower domain sizes with larger problems the average decrease in runtimes and maximum runtime improvement with case 1 reveals that ODE does leave room for a significant improved performance.

Although runtime improvements are more visibly pronounced with larger d values, overall, with the elimination of the right set of definitions, ODE can lead to significant performance gains between 10-100% and scalability. For example, with ODE we could find models for CODI using Paradox with ABoxes containing atleast 50 individuals and 200 relational assertions, whereas previously Paradox ran out of time with 30 individuals and 100 assertions.

7.4 Discussion and Conclusion

Through an experimental study with a set of spatial ontologies and the best available model finders we verified that the runtime of model finders (that do not employ predicate elimination) is actually closely correlated to that growth in the ontology's size measures. In that sense the work undertaken here goes further than previous studies that only compare performance between model finders without looking at which parameters affect the model finder's performance most. Using FOL-based definition elimination with solvers that do not perform their own predicate elimination and do FOL-based definition elimination, here with Paradox and Vampire, led to a more *consistent* improvement in model finding as opposed to iProver which exhibits very unpredictable results. With ODE, Paradox scaled to generate models for the RCC ontology with ABoxes with $d = 40$ and $r = 10$, which is a very significant

[15]Our experimental results are limited to two low values for $d = 10$, 15 – as solvers quickly run into intractability due to the hardness of the INCH ontology

improvement in performance compared to iProver, which became intractable on problems half this size.

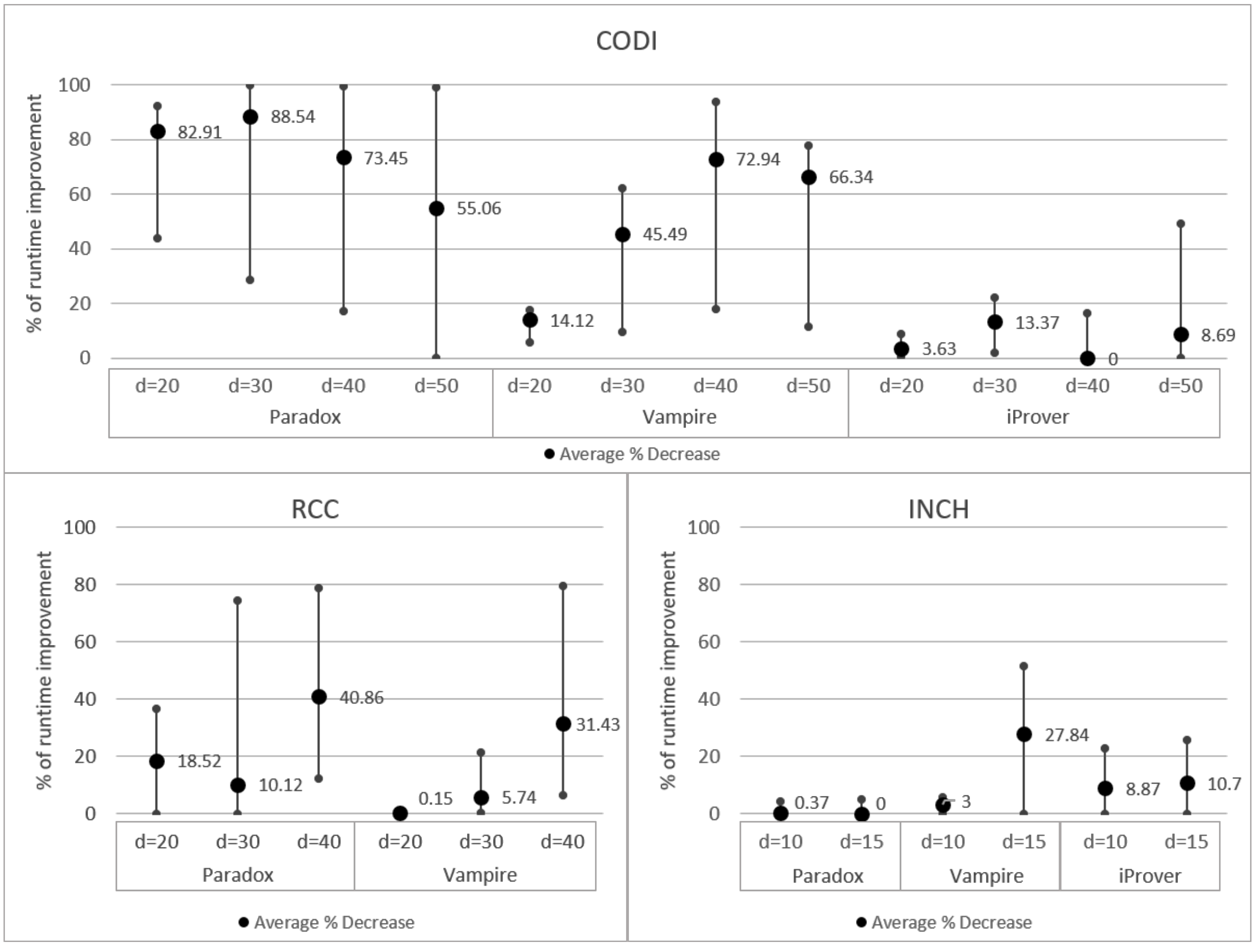


Figure 7.7: Reduction in low mean model finding time (y axis) for different domain sizes (x axis). The reduction is measured as a percentage of runtime decrease from the default case. The maximum decrease (represented by the small circle on the top of the high-low lines) represents the decrease calculated with the best case (uniformly determined across all (r-d)ABoxes).

We also presented results that show an improved performance of solvers when reasoning with medium-sized datasets. We found that with ODE we were able to solve examples that were previously intractable. We expect the experimental developments presented in this paper can provide some insights into specific preprocessing steps that can be inbuilt into

FOL model-finders. Further, in order to make ODE effective and efficient, the results address the following questions: (1) When should we activate ODE? (2) Which optionally defined predicates should we eliminate? The most satisfactory answers to these questions may depend on other techniques implemented in the solver, and problem hardness characteristics not studied in this work. Nevertheless, we want to argue that the general principle for guiding the implementation of ODE for any axiomatization to identify the *best* case, which is usually the case that minimizes the following measures the utmost: (1) number of predicates of highest arity to reduce P_v, (2) FOL-CNF clauses with high variable-density to reduce P_c, (2) FOL-CNF formula-width. Theoretical calculation of these measures can be used to develop a ODE preprocessing heuristic that can be implemented into ATPs in the future as extension of this dissertation.

CHAPTER 8

CONCLUSION

This dissertation presents two principal results that align with the overarching objectives highlighted in Section 1.2 in terms of integrated spatial reasoning. First we have focused on the representational aspects that merges qualitative and geometric spatial information to unify the two kinds of representations within a single framework, thereby addressing the specific objective O1 from Section 1.2.2. Secondly we develop a formal framework for size or complexity measures of ontologies with data addressing objective O2. Finally we investigate a FOL ontology preprocessing technique to improve the scalability of spatial reasoning such as consistency checking and query answering through model finding, thereby addressing objectives O3-O4. Here below, we present a summary of this dissertation and highlight the important contributions we have made to improve FOL-based spatial reasoning.

Simple Features Access (SFA-FOL): Currents trends in qualitative spatial reasoning include using spatial operators to compute qualitative relations between vector geometries in a spatial database, or using formal spatial ontologies to query over a set of geometric data assertions. Axiomatic representations enable reasoning consistent with common-sense reasoning [94] in varying degrees. However existing formalisms are limited in certain ways, and any one separate model is incapable of handling the mixture of real-world spatial data as they exist in GIS databases and non-geometric sources. Popular representations such as the RCC-8 and the 9-IM only model topological relations between objects of the same dimension, while 9-IM does not denote the dimensionality of shared region; the DE-9IM used in SFA cannot handle complex objects with holes and parts; Freska's Double cross calculus is limited to 2-D [102]; graph-based approaches [184] do not tie vector geometric concepts to qualitative relations. Some of these limitations are tackled in multi-dimensional mereotopologies such as CODIB and multi-dimensional RCC [154], but they still do not allow seamless integrated reasoning that combines data from geometric and non-geometric sources. On the other hand, works undertaken to integrate qualitative spatial reasoning over spatial geometries using reasoning tools such as Racer [274] and Pellet [258] or even spatial extensions of RDF and SPARQL, such as stSPARQL [170] or GeoSPARQL [220] use DL-based ontologies that lack the semantics available in FOL ontologies. In Chapter 4 we have developed a qualitatively augmented formalization of the Simple Features Access model in FOL as an extension of CODI and CODIB to tackle these limitations. The formalization presented in this chapter shows that geometric concepts (e.g. polygon) can be considered as specialization of qualitative concepts (e.g. ArealRegion) and all the qualitative relations apply equally to geometric and qualitative concepts. This ontology, SF-FOL, can now be used to ingest traditional geometric information that resides in spatial knowledge bases as well as qualitative information from any external non-geometric source and FOL automated reasoners can be used to reason over a mix of both.

Model Finding using FOL Spatial Ontologies: The formalization of SF-FOL serves as a unifying representation for integrated spatial reasoning. Specific reasoning tasks include theorem proving, consistency checking and query answering. We identified that much of prior work on FOL reasoning focuses on theorem proving tasks that are, while also theoretically intractable, comparatively easier than model finding. But even these works have mostly used axiomatizations with signatures that do not reflect the signatures of realistic domain and application ontologies, nor do they use any datasets in the reasoning. Even leading SAT-based ATPs have sophisticated mechanisms to handle theorem proving for mathematical axiomatizations and [97, 10], but poorly performed with our spatial model finding problems. Little work has been done to systematically test model finding with tools rarely successful or producing only very small, often trivial models that do not exceed 20 individuals. The exact root sources of the poor performance of model finding for FOL ontologies have also never been clearly investigated and quantified, thus preventing any progress on improving model finding with FOL ontologies. Despite the theoretical hardness of large SAT problems, practical SAT solvers have made strident progress in successfully scaling and solving large propositional logic problems. This is vastly attributed to formula simplification techniques, some of which have even been lifted to FOL ATPs. But FOL ontologies lead to a combinatorial increase in the size of their SAT problems with increasing sizes of the domain and terminology leading to a dramatic increase in the SAT search space. So, while these simplifications strategically and successfully reduce problem size and therefore search space, the magnitude of simplification still does not allow for scalable model finding for even moderate-sized datasets that are needed for simple reasoning tasks with FOL ontologies[1] . On the other hand, some of these simplifications are intrinsically computationally complex, for example identifying which clauses to remove is non-trivial [183]. Another driver for solver advancement is the experimental research effort towards understanding complexity of problems using benchmark problems. SAT benchmarks do not reflect the complexity and size of problems that arise from the translation of FOL ontologies with data to propositional logic. And ATP benchmarks, specifically the TPTP suite functions well for evaluating theorem provers but less so for model finders.

With this knowledge of state-of-the-art and limitations that impede extensible spatial reasoning, we have identified and studied specific measures that lead to intractability of FOL model finding. The contributions made in this regard are two-fold. First we have provided formal semantics for the TBox, ABox and different sets of optional definitions that can be removed from an FOL ontology. We have identified the number of predicates of the highest arity and domain size of the ABox in a FOL ontology, and the number of FOL-CNF clauses, variable-density and formula-width of its FOL-CNF translation as the attributes that have an outsized influence on the size and difficulty of the resulting SAT problems for model finding. Through theoretical calculations we have found these parameters contribute to the growth of the SAT problem specifically in terms of its number of propositional variables P_v and propositional clauses P_c. These are also the two size measures that correlate to runtime of solvers as we have found in our work. A large ontology signature, especially the set of

[1]Our experiments revealed that with Vampire - the best performing solver - tractable model finding is limited to domain size 40 in CODI and domain size 15 in INCH.

predicates of highest arity, exponentially increases the size of the SAT problem in terms of P_v with increasing domain size. This signature stems from the set of predicates in the axiomatization (TBox) and any additional predicates that get introduced from clausifying the TBox and the ABox. Many defined predicates in the TBox also contribute to a significant increase in P_c, due to the elimination of biconditionals during clausification. With this insight into the growth in size of a FOL ontology's SAT translation, we consequently define optional definition elimination (ODE) to syntactically simplify the FOL ontology by altering the DBox and ABox while preserving the structure of any possible models. ODE can prune the search space significantly dependent on the number of optionally defined predicates and their axiomatic simplicity, and is capable of significantly reducing the size of a problem by orders of magnitude, thus verifying our hypothesis that aggressive ODE on predicates of highest arity yields a significant reduction in the number of propositional variables.

By implementing ODE at different degrees, we compared calculated measures of FOL ontologies against practical hardness of model finding, in particular to understand whether the size of the SAT problem is a good indicator of practical hardness. This was accomplished through conducting comprehensive experiments on benchmark problems by varying three parameters: signature of the ontology, domain size and number of relational assertions in the ABox. To the best of our knowledge, no such systemic model finding experiments have previously been reported on, and is an important step that actually establishes a strong correlation correlation between the two that informs future work on automatically preprocessing FOL ontologies for improved model finding. The experiments on the benchmark ontologies revealed that with ODE we were able to achieve speedups upto at least 10% and as high as 99%, and even improving scalability of model finding to domain sizes that were previously intractable. This confirms our hypothesis that removing optional definitions from the TBox can significantly improve FOL model finding performance, and demonstrating the feasibility of model finding with mid-sized spatial data sets. To further complement the empirical evaluation, we identified that formula-width is another important measure for estimating how difficult the SAT problem that results from an FOL ontology may be, and a good indicator for determining whether to eliminate a certain definition or not. Applying ODE to ontologies with nested defined predicates not only leads to an exponential increase in the number of clauses, but also formulas with large formula-width, and thus does not justify the most aggressive definition elimination that is theoretically possible. The preprocessing algorithm must therefore be optimized to select the level upto which ODE is maximally efficient.

We have found ODE to at least alleviate the problem of intractabilty encountered with increasing domain sizes during model finding and enable reasoning with more reasonably sized, though still relatively small in today's big data expectations, samples from datasets. But using ODE is to syntactically alter an ontology to improve model finding performance by potentially decreasing the runtime by orders of magnitudes and, as a more important effect, allowing to successfully verify ontologies against data sets with larger domain sizes is a very important finding.

Preprocessing is crucial when dealing with large ontologies especially in the presence of data. Given a FOL ontology, the goal of ODE is to translate it to an equisatisfiable variant with a smaller signature that minimizes P_v and P_c in its SAT translation while avoiding adverse consequences from applying ODE aggressively. ODE aims to reduce the number of predicates in the FOL-CNF formula, because each (a-nary) predicate leads to d^a propositional variables for domain d. But, since recursive ODE increases the possibility of creating longer FOL formulas, it is important to limit ODE to predicates whose elimination does not result in alarmingly long formulas that in turn lead to longer FOL-CNF formulas. The measures that we identify (on the ontology and its FOL-CNF formula) can be specifically used for a heuristic analysis – to automatically calculate the resulting P_v, while not significantly increasing P_c with and without definition elimination (e.g. take one definition, see whether it should be eliminated, then move on to the next definition until we have a decision for all predicates in the DBox. ODE as presented here advances on definition inlining outlined in [229] that also uses contextual information to inline definitions and reduce the size of a problem's signature. Our experimental results demonstrate performance gains from ODE are quite significant over any inlining simplification already implemented in Vampire. Other predicate elimination procedures that reduce the FOL ontology signature resembling our implementation of ODE are PPE and UDE [161] implemented in Vampire, however even with these simplifications our experiments revealed intractability on problems with more than 920,000 propositional variables and 160,0000 propositional clauses[2] . Our experiments on the sample ontologies show that through ODE the reduction in P_v and P_c is as high as 83% and 72%. For Paradox and Vampire, this leads to a speed-up model finding time by at least 3 and 1 orders of magnitude, respectively, across the ABoxes of different sizes. More importantly, ODE enables tractable model finding on larger problems on which solvers with standard simplification procedures failed. For example, by applying ODE to CODI, the size of models that could be found by Paradox increased from domain size 30 to domain sizes of 120 and beyond. Unlike many simplifications such as identifying blocked clauses for elimination, which in itself is NP hard or others which are at best polynomial [163], ODE can be implemented without compromising efficiency, on the FOL problem before its translation to FOL-CNF or propositional-CNF.

Additionally, through correlation analysis we verify that our results are consistent with our general hypothesis that the difficulty of model finding is determined by the overall size of the signature. In particular, the number of propositional variables might be a more precise indicator of practical hardness than the number of propositional clauses or their width, in the sense that the latter two measures give too optimistic estimates for formulas which have very low number of clauses or width but which might nevertheless be hard for solvers to solve in practice. We believe that ODE when combined with efficient heuristics (that can be guided by our results) is a promising FOL-simplification paradigm for model finding with

[2]Resembling case 13 in CODI for $d = 50$ and $r = 5$. This is the default case including all the defined predicates with no ODE performed - thereby allowing Vampire to perform any default simplifications such as inlining, PPE and UDE.

complex axiomatizations and moderately sized datasets. In addition to typical reasoning tasks, this kind of scalability will aid in the external verification of ontologies, specifically spatial ontologies and reasoning with them (as evident from our model finding results using CODI, RCC and INCH against small real vector datasets), identifying suitable spatial background theories for a dataset and also more generically in data repairing.

8.1 Future Work

A conspicuous point for future work, and also as a next step of this work would be to include the developments of our findings into an automated heuristic preprocessing tool and verify the implementation with much larger and diverse (non-geospatial) set of ontologies to see how broadly useful it is. But a challenge that ensues is that such a task will be limited to ontologies for which real data is easily accessible. Another concern is that there are still many open questions about what makes some solvers tick. From our experimental findings, iProver did not gain as much benefit from ODE as Vampire and Paradox.

Another interesting line of investigation would be to study the implications of ODE on non-MACE systems like Darwin-style and SMT solvers or even theorem provers. The flattening transformation in Paradox generates one $n+1$-ary predicate symbol in the FOL-CNF translation for each n-ary function symbol in the FOL formula. But Darwin, replaces all n-ary function symbols with one $n+2$-ary function symbol – which means Darwin adds fewer Skolem predicates from clausification compared to Paradox, but some of these predicates could also be of an arity higher than any predicate generated by Paradox. This kind of a *meta-modeling* approach yields a more compact clause set, but also operates in the function-free logic fragment, where the growth in problem size is much slower than propositional problems. We therefore expect ODE could lead to more significant performance gains with Darwin-style solvers over MACE-style solvers.

A certain extent of redundancy is generally thought to improve solver performance. For example, several works find lemmas and redundant axioms to make theorem proving problems easier [55], redundant clauses boost solver performance [156], and constrainedness (in terms of clause density) affect problem hardness [53, 206, 117]. Likewise our experiments reveal constrainedness of ABox assertions (ratio of d to r), specifically in RCC and INCH influence the performance of Paradox and Vampire (discussed in Section 7.2.1). We plan to investigate the influence of this parameter on model finding to investigate the question whether sub-models, or new assertions proved from smaller/easier problems, can be added to a more difficultly constrained problem to scale reasoning for larger domain sizes.

Splitting is a reduction technique to minimize the number of variables v in clauses[3] , but introduces additional clauses and more importantly the addition of new predicate symbols [56]. However our theoretical results reveal that with optimal ODE, v is already reduced, and since P_v has an outsized impact on hardness compared to P_c, it would be worthwhile to investigate the interaction between the two simplification techniques – if turning off splitting when using ODE can further improve model finding performance.

[3]Since the number of propositional instantiations for each FOL-CNF clause in a formula is exponential in the number of variables in the clause.

REFERENCES

[1] https://github.com/gruninger/colore/mereotopology, accessed 04/2020.

[2] https://github.com/gruninger/colore/inch, accessed 04/2020.

[3] Emmanuel Abbe and Andrea Montanari. On the Concentration of the Number of Solutions of Random Satisfiability Formulas. *Random Structures & Algorithms*, 45(3):362–382, 2014.

[4] Dimitris Achlioptas. Lower Bounds for Random 3-SAT via Differential Equations. *Theoretical Computer Science*, 265(1-2):159–185, 2001.

[5] Dimitris Achlioptas and Federico Ricci-Tersenghi. On the Solution-Space Geometry of Random Constraint Satisfaction Problems. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 130–139. ACM, 2006.

[6] Eric Allender, Shiteng Chen, Tiancheng Lou, Periklis Papakonstantinou, and Bangsheng Tang. Width-parameterized SAT: Time-space tradeoffs. 2014.

[7] Fadi A Aloul, Arathi Ramani, Igor L Markov, and Karem A Sakallah. Solving Difficult SAT Instances in the Presence of Symmetry. In *Proceedings of the 39th annual Design Automation Conference*, pages 731–736. ACM, 2002.

[8] Teresa Alsinet, Felip Manya, and Jordi Planes. A Max-SAT Solver with Lazy Data Structures. In *Ibero-American Conference on Artificial Intelligence*, pages 334–342. Springer, 2004.

[9] Marcelo Arenas and Jorge Pérez. Querying Semantic Web Data with SPARQL. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 305–316, 2011.

[10] Rob Arthan and Paulo Oliva. (Dual) Hoops have Unique Halving. In *Automated Reasoning and Mathematics*, pages 165–180. Springer, 2013.

[11] Gilles Audemard and Laurent Simon. Glucose: a Solver that Predicts Learnt Clauses Quality. *SAT Competition*, pages 7–8, 2009.

[12] Fahiem Bacchus. Enhancing Davis Putnam with Extended Binary Clause Reasoning. *AAAI/IAAI*, 2002:613–619, 2002.

[13] Fahiem Bacchus, Shannon Dalmao, and Toniann Pitassi. Solving #SAT and Bayesian Inference with Backtracking Search. *Journal of Artificial Intelligence Research*, 34:391–442, 2009.

[14] Fahiem Bacchus and Jonathan Winter. Effective Preprocessing with Hyper-Resolution and Equality Reduction. In *International conference on theory and applications of satisfiability testing*, pages 341–355. Springer, 2003.

[15] Clark W Barrett, David L Dill, and Aaron Stump. Checking Satisfiability of First-Order Formulas by Incremental Translation to SAT. In *International Conference on Computer Aided Verification*, pages 236–249. Springer, 2002.

[16] Jon Barwise, Solomon Feferman, and Solomon Feferman. *Model-theoretic logics*, volume 8. Cambridge University Press, 2017.

[17] John Bateman and Scott Farrar. Spatial Ontology Baseline, D2, I1-[OntoSpace]. Technical report, Collaborative Research Center for Spatial Cognition, University of Bremen, 2006.

[18] Sotiris Batsakis and Euripides GM Petrakis. SOWL: Spatio-Temporal Representation, Reasoning and Querying over the Semantic Web. In *Proceedings of the 6th international conference on semantic systems*, pages 1–9, 2010.

[19] Ringo Baumann and Heinrich Herre. The Axiomatic Foundation of Space in GFO. *Applied Ontology*, 2011. submitted.

[20] Ringo Baumann, Frank Loebe, and Heinrich Herre. Towards an Ontology of Space for GFO. In *Conf. on Formal Ontology in Inf. Systems (FOIS-16)*, pages 53–66, 2016.

[21] Peter Baumgartner. A First-Order Davis-Putnam-Logemann-Loveland Procedure. In *IJCAR2001, International Joint Conference on Artificial Intelligence*, 2002.

[22] Peter Baumgartner, Norbert Eisinger, and Ulrich Furbach. A Confluent Connection Calculus. In *International Conference on Automated Deduction*, pages 329–343. Springer, 1999.

[23] Peter Baumgartner, Alexander Fuchs, Hans De Nivelle, and Cesare Tinelli. Computing finite Models by Reduction to Function-Free Clause Logic. *Journal of Applied Logic*, 7(1):58–74, 2009.

[24] Peter Baumgartner, Alexander Fuchs, and Cesare Tinelli. Darwin: A Theorem Prover for the Model Evolution Calculus. In *IJCAR Workshop on Empirically Successful First Order Reasoning (ESFOR (aka S4)), Electronic Notes in Theoretical Computer Science*, page 191. Citeseer, 2004.

[25] Peter Baumgartner, Björn Pelzer, and Cesare Tinelli. Model Evolution with Equality-Revised and Implemented. *Journal of Symbolic Computation*, 47(9):1011–1045, 2012.

[26] Peter Baumgartner and Cesare Tinelli. The Model Evolution Calculus. In *International Conference on Automated Deduction*, pages 350–364. Springer, 2003.

[27] Roberto J Bayardo Jr and Robert Schrag. Using CSP Look-Back Techniques to solve real-world SAT instances. In *Aaai/iaai*, pages 203–208. Providence, RI, 1997.

[28] Paul Beame, Henry Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *Journal of Artificial Intelligence Research*, 22:319–351, 2004.

[29] Mordechai Ben-Ari. *Mathematical logic for computer science*. Springer Science & Business Media, 2012.

[30] Konstantina Bereta, Guohui Xiao, Manolis Koubarakis, Martina Hodrius, Conrad Bielski, and Gunter Zeug. Ontop-spatial: Geospatial Data Integration using GeoSPARQL-to-SQL translation. In *Proceedings of the 15th International Semantic Web Conference, Posters & Demonstrations Track (ISWC). Available at: http://ceur-ws. org*, volume 1690, 2016.

[31] Jeremias Berg and Matti Järvisalo. SAT-Based Approaches to Treewidth Computation: An evaluation. In *2014 IEEE 26th International Conference on Tools with Artificial Intelligence*, pages 328–335. IEEE, 2014.

[32] Evert W. Beth. On Padoa's Method in the Theory of Definition. *Indagationes Math.*, 15:330–339, 1953.

[33] Armin Biere. Resolve and Expand. In *International conference on theory and applications of satisfiability testing*, pages 59–70. Springer, 2004.

[34] Armin Biere. Preprocessing and Inprocessing Techniques in SAT. In *Haifa Verification Conference*, volume 1, 2011.

[35] Armin Biere. Splatz, Lingeling, PLingeling, Treengeling, Yalsat entering the SAT Competition 2016. *Proc. of SAT Competition*, pages 44–45, 2016.

[36] Armin Biere, Alessandro Cimatti, Edmund M Clarke, Ofer Strichman, and Yunshan Zhu. Bounded model checking. *Handbook of satisfiability*, 185(99):457–481, 2009.

[37] Armin Biere, Marijn Heule, and Hans van Maaren. *Chapter 5: Look-Ahead Based SAT Solvers. In Handbook of satisfiability*, volume 185. IOS press, 2009.

[38] Per Bjesse, James Kukula, Robert Damiano, Ted Stanion, and Yunshan Zhu. Guiding SAT Diagnosis with Tree Decompositions. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 315–329. Springer, 2003.

[39] Per Bjesse, Tim Leonard, and Abdel Mokkedem. Finding Bugs in an Alpha Microprocessor usingSsatisfiability Solvers. In *International Conference on Computer Aided Verification*, pages 454–464. Springer, 2001.

[40] Patrick Blackburn and Johan Bos. Representation and Inference for Natural Language. *A first course in computational semantics. CSLI*, 2005.

[41] Ivan Bocic and Tevfik Bultan. Efficient Data Model Verification with Many-Sorted Logic (T). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 42–52. IEEE, 2015.

[42] Maria Paola Bonacina. On Theorem Proving for Program Checking: Historical Perspective and Recent Developments. In *Proceedings of the 12th international ACM SIGPLAN symposium on Principles and practice of declarative programming*, pages 1–12, 2010.

[43] Lucas Bordeaux, Youssef Hamadi, and Lintao Zhang. Propositional Satisfiability and Constraint Programming: A comparative survey. *ACM Computing Surveys (CSUR)*, 38(4):12, 2006.

[44] Stefano Borgo, Nicola Guarino, and Claudio Masolo. A Pointless Theory of Space based on Strong Connection and Congruence. In *KR'96: Principles of Knowledge Representation and Reasoning*, pages 220–229, 1996.

[45] Stefano Borgo and Claudio Masolo. Full Mereogeometries. *Rev. Symb. Logic*, 3(4):521–567, 2010.

[46] Johan Bos. Exploring Model Building for Natural Language Understanding. In *Proc. ICoS*, volume 4, 2003.

[47] Ronen I Brafman. A Simplifier for Propositional Formulas with many Binary Clauses. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(1):52–59, 2004.

[48] Alfredo Braunstein, Marc Mézard, and Riccardo Zecchina. Survey Propagation: An algorithm for satisfiability. *Random Structures & Algorithms*, 27(2):201–226, 2005.

[49] Boyan Brodaric and Torsten Hahmann. Towards a Foundational Hydro Ontology for Water Data Interoperability. In *Int. Conf. on Hydroinformatics (HIC-2014)*, 2014.

[50] Andrei Z Broder, Alan M Frieze, and Eli Upfal. On the Satisfiability and Maximum Satisfiability of Random 3-CNF formulas. In *SODA*, volume 93, pages 322–330, 1993.

[51] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. A Duality between Clause Width and Clause Density for SAT. In *21st Annual IEEE Conference on Computational Complexity (CCC'06)*, pages 7–pp. IEEE, 2006.

[52] Roberto Casati and Achille C. Varzi. *Parts and Places.* MIT Press, 1999.

[53] Peter C Cheeseman, Bob Kanefsky, and William M Taylor. Where the really hard problems are. In *IJCAI*, volume 91, pages 331–337, 1991.

[54] Jingchao Chen. Fast Blocked Clause Decomposition with High Quality. *arXiv preprint arXiv:1507.00459*, 2015.

[55] Koen Claessen, Reiner Hähnle, and Johan Mårtensson. Verification of Hardware Systems with First-Order Logic. In *Proceedings of the CADE-18 Workshop-Problem and Problem Sets for ATP*, number 02/10, 2002.

[56] Koen Claessen and Niklas Sörensson. New Techniques that Improve MACE-style Finite Model Building. In *Workshop on Model Computation at CADE 2003*, 2003.

[57] Eliseo Clementini and Paolino Di Felice. A Comparison of Methods for Representing Topological Relationships. *Information sciences-applications*, 3(3):149–178, 1995.

[58] Eliseo Clementini, Paolino Di Felice, and Daniel Hernández. Qualitative Representation of Positional Information. *Artificial intelligence*, 95(2):317–356, 1997.

[59] Eliseo Clementini, Paolino Di Felice, and Peter Van Oosterom. A small set of formal topological relationships suitable for end-user interaction. In *International Symposium on Spatial Databases*, pages 277–295. Springer, 1993.

[60] Eliseo Clementini and Paolino Di Felice. A Model for Representing Topological Relationships between Complex Geometric Features in Spatial Databases. *Inf. Sci.*, 90(1):121–136, 1996.

[61] Cristian Coarfa, Demetrios D Demopoulos, Alfonso San Miguel Aguirre, Devika Subramanian, and Moshe Y Vardi. Random 3-sat: The plot thickens. In *International Conference on Principles and Practice of Constraint Programming*, pages 143–159. Springer, 2000.

[62] Anthony G Cohn, Brandon Bennett, John Gooday, and Nicholas Mark Gotts. Qualitative Spatial Representation and Reasoning with the Region Connection Calculus. *GeoInformatica*, 1(3):275–316, 1997.

[63] Anthony G. Cohn, Brandon Bennett, John M. Gooday, and Nicholas M. Gotts. Representing and Reasoning with Qualitative Spatial Relations about Regions. In Oliviero Stock, editor, *Spatial and Temporal Reasoning*, pages 97–134. Kluwer, 1997.

[64] Anthony G. Cohn and Jochen Renz. Qualitative Spatial Representation and Reasoning. In F. van Harmelen, V. Lifschitz, and B. Porter, editors, *Handbook of Knowledge Representation*. Elsevier, 2008.

[65] Anthony G Cohn and Achille C Varzi. Mereotopological connection. *Journal of Philosophical Logic*, 32(4):357–390, 2003.

[66] Stephen A Cook. The Complexity of Theorem-Proving Procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971.

[67] Bruno Courcelle, Johann A. Makowsky, and Udi Rotics. On the Fixed Parameter Complexity of Graph Enumeration Problems Definable in Monadic Second-Order Logic. *Discrete Applied Mathematics*, 108(1-2):23–52, 2001.

[68] Kenny R Coventry and Simon C Garrod. *Saying, seeing and acting: The psychological semantics of spatial prepositions*. Psychology Press, 2004.

[69] Simon Cox. Observations and measurements. *Open Geospatial Consortium Best Practices Document. Open Geospatial Consortium*, page 21, 2006.

[70] James M Crawford and Larry D Auton. Experimental results on the crossover point in satisfiability problems. In *AAAI*, volume 93, pages 21–27. Citeseer, 1993.

[71] James M Crawford and Larry D Auton. Experimental Results on the Crossover Point in Random 3-SAT. *Artificial intelligence*, 81(1-2):31–57, 1996.

[72] Cycorp Inc. OpenCyc Selected Vocabulary and Upper Ontology, Spatial Relations. `http://www.cyc.com/cycdoc/vocab/spatial-vocab.html`, 02 2012.

[73] Evgeny Dantsin and Alexander Wolpert. On Moderately Exponential Time for SAT. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 313–325. Springer, 2010.

[74] Martin Davis, George Logemann, and Donald Loveland. A Machine Program for Theorem-Proving. *Communications of the ACM*, 5(7):394–397, 1962.

[75] Martin Davis and Hilary Putnam. A Computing Procedure for Quantification Theory. *Journal of the ACM (JACM)*, 7(3):201–215, 1960.

[76] Giuseppe De Giacomo and Maurizio Lenzerini. TBox and ABox Reasoning in Expressive Description Logics. In *KR'96: Principles of Knowledge Representation and Reasoning*, pages 316–327, 1996.

[77] Rina Dechter and Judea Pearl. Tree Clustering for Constraint Networks. *Artificial Intelligence*, 38(3):353–366, 1989.

[78] Nachum Dershowitz, Ziyad Hanna, and Alexander Nadel. A Clause-Based Heuristic for SAT Solvers. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 46–60. Springer, 2005.

[79] Lyndon Drake and Alan Frisch. The Interaction Between Inference and Branching Heuristics. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 370–382. Springer, 2003.

[80] Lyndon Drake, Alan Frisch, Inês Lynce, JP Marques-Silva, and Toby Walsh. Comparing SAT Preprocessing Techniques. 2002.

[81] Heshan Du, Natasha Alechina, Kristin Stock, and Michael Jackson. The Logic of NEAR and FAR. In *International Conference on Spatial Information Theory*, pages 475–494. Springer, 2013.

[82] André Duarte and Konstantin Korovin. Experimenting with Superposition in iProver. In *WORKSHOP 2019*, page 27, 2019.

[83] Olivier Dubois. Sat versus unsat. *Cliques, Coloring, and Satisfiability: Series in Discrete Mathematics and Theoretical Computer Sicence*, 26:415–436, 1996.

[84] Olivier Dubois, Yacine Boufkhad, and Jacques Mandler. Typical Random 3-SAT Formulae and the Satisfiability Threshold. *arXiv preprint cs/0211036*, 2002.

[85] Jeffrey M Dudek, Kuldeep S Meel, and Moshe Y Vardi. The Hard Problems are almost everywhere for Random CNF-XOR Formulas. *arXiv preprint arXiv:1710.06378*, 2017.

[86] Vincent Dugat, Pierre Gambarotto, and Yannick Larvor. Qualitative Theory of Shape and Orientation. In *Proc. of the 16th Int. Joint Conference on Artificial Intelligence (IJCAI'99), Stockolm, Sweden*, pages 45–53, 1999.

[87] Vijay Durairaj and Priyank Kalla. Guiding CNF-SAT Search via Efficient Constraint Partitioning. In *Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, pages 498–501. IEEE Computer Society, 2004.

[88] Niklas Eén and Armin Biere. Effective Preprocessing in SAT through Variable and Clause Elimination. In *International conference on theory and applications of satisfiability testing*, pages 61–75. Springer, 2005.

[89] Niklas Eén and Niklas Sörensson. An Extensible SAT-solver. In *SAT*, 2003.

[90] Niklas Een and Niklas Sorensson. The MINISAT Page. `http://minisat.se`, 2016.

[91] Max J. Egenhofer. Reasoning about Binary Topological Relations. In *Symp. on Large Spatial Databases (SSD'91)*, LNCS 525, pages 141–160. Springer, 1991.

[92] Max J. Egenhofer and John Herring. Categorizing Binary Topological Relations between Regions, Lines, and Points in Geographic Databases. Technical report, Department of Surveying Engineering, Univ. of Maine, 1991.

[93] Mohamed El Halaby. On the Computational Complexity of MaxSAT. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 23, page 34, 2016.

[94] Zoe Falomir. Towards a Qualitative Descriptor for Paper Folding Reasoning. In *Proc. of the 29th International Workshop on Qualitative Reasoning (QR?16)*, 2016.

[95] Katalin Fazekas, Armin Biere, and Christoph Scholl. Incremental Inprocessing in SAT Solving. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 136–154. Springer, 2019.

[96] Giorgio De Felice, Paolo Fogliaroni, and Jan Oliver Wallgrün. A Hybrid Geometric-Qualitative Spatial Reasoning System and its Application in GIS. In *Conf. on Spatial Inf. Theory (COSIT-09)*, 2009.

[97] Branden Fitelson. Gibbards Collapse Theorem for the Indicative Conditional: An Axiomatic Approach. In *Automated Reasoning and Mathematics*, pages 181–188. Springer, 2013.

[98] Jörg Flum. RG Downey and MR Fellows. Parameterized Complexity. Monographs in Computer Science. Springer, New York, Berlin, and Heidelberg, 1999, xv+ 533 pp. *Bulletin of Symbolic Logic*, 8(4):528–529, 2002.

[99] Paolo Fogliaroni, Paul Weiser, and Heidelinde Hobel. Qualitative Spatial Configuration Search. *Spatial Cognition & Computation*, 16(4):272–300, 2016.

[100] Andreas Folkler. Automated Theorem Proving: Resolution vs. Tableaux, 2002.

[101] Jon William Freeman. *Improvements to Propositional Satisfiability Search Algorithms*. PhD thesis, University of Pennsylvania Philadelphia, PA, 1995.

[102] Christian Freksa. Using Orientation Information for Qualitative Spatial Reasoning. In *Theories and methods of spatio-temporal reasoning in geographic space*, pages 162–178. Springer, 1992.

[103] Eugene C Freuder. A Sufficient Condition for Backtrack-Bounded Search. *Journal of the ACM (JACM)*, 32(4):755–761, 1985.

[104] Ehud Friedgut, Jean Bourgain, et al. Sharp Thresholds of Graph Properties, and the k-SAT Problem. *Journal of the American mathematical Society*, 12(4):1017–1054, 1999.

[105] Alan Frieze and Stephen Suen. Analysis of Two Simple Heuristics on a Random Instance of k-SAT. *Journal of Algorithms*, 20(2):312–355, 1996.

[106] Nicola Galesi and Oliver Kullmann. Polynomial Time SAT Decision, Hypergraph Transversals and the Hermitian Rank. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 89–104. Springer, 2004.

[107] Anthony Galton. Taking Dimension Seriously in Qualitative Spatial Reasoning. In *Europ.Conf. on Artif. Intell. (ECAI-96)*, pages 501–505, 1996.

[108] Robert Ganian, Neha Lodha, Sebastian Ordyniak, and Stefan Szeider. SAT-Encodings for Treecut Width and Treedepth. In *2019 Proceedings of the Twenty-First Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 117–129. SIAM, 2019.

[109] Harald Ganzinger and Konstantin Korovin. New Directions in Instantiation-Based Theorem Proving. In *18th Annual IEEE Symposium of Logic in Computer Science, 2003. Proceedings.*, pages 55–64. IEEE, 2003.

[110] Michael R Garey and David S Johnson. *Computers and intractability*, volume 29. wh freeman New York, 2002.

[111] Martin Gebser, Tomi Janhunen, Roland Kaminski, Torsten Schaub, and Shahab Tasharrofi. Writing Declarative Specifications for Clauses. In *European Conference on Logics in Artificial Intelligence*, pages 256–271. Springer, 2016.

[112] Roman Gershman and Ofer Strichman. Cost-Effective Hyper-Resolution for Preprocessing CNF Formulas. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 423–429. Springer, 2005.

[113] Arup Kumar Ghosh. Speeding up SAT Solver by Exploring CNF Symmetries: Revisited. *arXiv preprint arXiv:1102.0230*, 2011.

[114] Enrico Giunchiglia, Marco Maratea, and Armando Tacchella. Dependent and Independent Variables in Propositional Satisfiability. In *European Workshop on Logics in Artificial Intelligence*, pages 296–307. Springer, 2002.

[115] Evgueni Goldberg and Yakov Novikov. Verification of Proofs of Unsatisfiability for CNF Formulas. In *Proceedings of the conference on Design, Automation and Test in Europe-Volume 1*, page 10886. IEEE Computer Society, 2003.

[116] Carla P Gomes, Henry Kautz, Ashish Sabharwal, and Bart Selman. Satisfiability Solvers. *Foundations of Artificial Intelligence*, 3:89–134, 2008.

[117] Carla P Gomes and Bart Selman. Computational Science: Can get Satisfaction. *Nature*, 435(7043):751, 2005.

[118] Nicholas Mark Gotts. Formalising Commonsense Topology: The INCH Calculus. In *Proc. Fourth International Symposium on Artificial Intelligence and Mathematics*, 1996.

[119] R Goyal and Max J Egenhofer. The Direction-Relation Matrix: A Representation for Directions Relations between Extended Spatial Objects. *the annual assembly and the summer retreat of University Consortium for Geographic Information Systems Science*, 3:95–102, 1997.

[120] Pierre Grenon. BFO in a nutshell: A Bi-Categorical Axiomatization of BFO and comparison with DOLCE. Technical report, Leipzig University, Institute of Formal Ontology and Medical Information Science (IFOMIS), 2003.

[121] Pierre Grenon and Barry Smith. SNAP and SPAN: Towards Dynamic Spatial Ontology. *J. Spat. Cogn. Comput.*, 4(1):69–104, 2004.

[122] Orna Grumberg, Assaf Schuster, and Avi Yadgar. Memory Efficient All-Solutions SAT Solver and its Application for Reachability Analysis. In *International Conference on Formal Methods in Computer-Aided Design*, pages 275–289. Springer, 2004.

[123] Michael Grüninger. Ontology of the Process Specification Language. In *Handbook on ontologies*, pages 575–592. Springer, 2004.

[124] Michael Grüninger, Katy Atefi, and Mark S Fox. Ontologies to Support Process Integration in Enterprise Engineering. *Computational & Mathematical Organization Theory*, 6(4):381–394, 2000.

[125] Yuri Gurevich and David G Mitchell. A SAT Solver Primer. 2005.

[126] Djamal Habet, Lionel Paris, and Cyril Terrioux. A Tree Decomposition Based Approach to Solve Structured SAT Instances. In *2009 21st IEEE International Conference on Tools with Artificial Intelligence*, pages 115–122. IEEE, 2009.

[127] Torsten Hahmann. *A Reconciliation of Logical Representations of Space: from Multidimensional Mereotopology to Geometry*. PhD thesis, Univ. of Toronto, 2013.

[128] Torsten Hahmann. On Decomposition Operations in a Theory of Multidimensional Qualitative Space. In *FOIS*, pages 173–186, 2018.

[129] Torsten Hahmann and Michael Grüninger. Multidimensional Mereotopology with Betweenness. In *Int. Joint Conf. on Artif. Intell. (IJCAI-11)*, pages 906–911, 2011.

[130] Torsten Hahmann and Michael Grüninger. A Naïve Theory of Dimension for Qualitative Spatial Relations. In *Symp. on Logical Formalizations of Commonsense Reasoning (CommonSense 2011)*. AAAI Press, 2011.

[131] Torsten Hahmann and Michael Grüninger. A Theory of Multidimensional Qualitative Space: Semantic integration of spatial theories that distinguish interior from boundary contact (extended abstract). In *Proc. of the Int. Conference on Spatial Information Theory (COSIT 2011), Belfast, Maine, September 12-16, 2011*, 2011.

[132] Torsten Hahmann and Michael Grüninger. Region-Based Theories of Space: Mereotopology and Beyond. In *Qualitative spatio-temporal representation and reasoning: Trends and future directions*, pages 1–62. IGI Global, 2012.

[133] Torsten Hahmann, Shirly Stephen, and Boyan Brodaric. Semantically Refining the GWML2 with the Help of a Reference Ontology. In *Conf. on Geographic Inf. Sci. (GIScience)*, 2016.

[134] Youssef Hamadi, Saïd Jabbour, and Lakhdar Sais. Learning for Dynamic Subsumption. *International Journal on Artificial Intelligence Tools*, 19(04):511–529, 2010.

[135] Hyojung Han and Fabio Somenzi. On-the-fly Clause Improvement. In *International conference on theory and applications of satisfiability testing*, pages 209–222. Springer, 2009.

[136] Brian Hayes. Computing science: Can't get no satisfaction. *American scientist*, 85(2):108–112, 1997.

[137] Julio Cesar Lopez Hernandez and Konstantin Korovin. Towards an Under-Approximation Abstraction-Refinement for Reasoning with Large Theories. In *WORKSHOP 2019*, page 3, 2019.

[138] J Herring. OpenGIS Implementation Standard for Geographic information - Simple Feature Access-Part 1: Common architecture, 2011.

[139] Marijn Heule, Matti Järvisalo, and Armin Biere. Clause Elimination Procedures for CNF Formulas. In *International Conference on Logic for Programming Artificial Intelligence and Reasoning*, pages 357–371. Springer, 2010.

[140] Marijn Heule, Matti Järvisalo, and Armin Biere. Covered Clause Elimination. *arXiv preprint arXiv:1011.5202*, 2010.

[141] Marijn Heule, Matti Järvisalo, Florian Lonsing, Martina Seidl, and Armin Biere. Clause Elimination for SAT and QSAT. *Journal of Artificial Intelligence Research*, 53:127–168, 2015.

[142] Marijn JH Heule and Armin Biere. Blocked Clause Decomposition. In *International Conference on Logic for Programming Artificial Intelligence and Reasoning*, pages 423–438. Springer, 2013.

[143] Marijn JH Heule, Matti Järvisalo, and Armin Biere. Efficient CNF Simplification Based on Binary Implication Graphs. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 201–215. Springer, 2011.

[144] Marijn JH Heule and Stefan Szeider. A SAT Approach to Clique-Width. *ACM Transactions on Computational Logic (TOCL)*, 16(3):24, 2015.

[145] Timothy L Hinrichs and Michael R Genesereth. *Extensional Reasoning*. Citeseer, 2008.

[146] Krystof Hoder, Zurab Khasidashvili, Konstantin Korovin, and Andrei Voronkov. Preprocessing Techniques for First-Order Clausification. In *12th Intern. Conf. on Formal Methods in Computer-Aided Design (FMCAD 2012)*, pages 44–51. IEEE, 2012.

[147] Holger H Hoos. On the Run-Time Behaviour of Stochastic Local Search Algorithms for SAT. In *AAAI/IAAI*, pages 661–666, 1999.

[148] Holger H Hoos and Thomas Stützle. Local Search Algorithms for SAT: An Empirical Evaluation. *Journal of Automated Reasoning*, 24(4):421–481, 2000.

[149] Holger H Hoos and Thomas Stützle. SATLIB: An Online Resource for Research on SAT. *Sat*, 2000:283–292, 2000.

[150] Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Reasoning with Individuals for the Description Logic SHIQ. In *CADE-2000*, pages 482–496. Springer, 2000.

[151] International Electrotechnical Commission (ISO/IEC). ISO 19125:2004 geographic Information – Simple Feature Access, 2004.

[152] International Electrotechnical Commission (ISO/IEC). ISO 19136:2007 Geographic Information – Geography Markup Language (GML), 2007.

[153] Kazuo Iwama and Kazuya Takaki. Satisfiability of 3-CNF Formulas with Small Clause/Variable Ratio. *American Mathematical Society, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 35:315–333, 1997.

[154] Azadeh Izadi, Kristin M Stock, and Hans W Guesgen. Multidimensional Region Connection Calculus. *Unpublished Paper, Association for the Advancement of Artifical Intelligence*, 2017.

[155] Matti Järvisalo, Armin Biere, and Marijn Heule. Blocked Clause Elimination. In *International conference on tools and algorithms for the construction and analysis of systems*, pages 129–144. Springer, 2010.

[156] Matti Järvisalo, Marijn JH Heule, and Armin Biere. Inprocessing Rules. In *International Joint Conference on Automated Reasoning*, pages 355–370. Springer, 2012.

[157] HoonSang Jin and Fabio Somenzi. An Incremental Algorithm to Check Satisfiability for Bounded Model Checking. *Electronic Notes in Theoretical Computer Science*, 119(2):51–65, 2005.

[158] Peter Jonsson, Victor Lagerkvist, Gustav Nordh, and Bruno Zanuttini. Complexity of SAT Problems, Clone Theory and the Exponential Time Hypothesis. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1264–1277. Society for Industrial and Applied Mathematics, 2013.

[159] Soumya C Kambhampati and Thomas Liu. Phase Transition and Network Structure in Realistic SAT Problems. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[160] Hadi Katebi, Karem A Sakallah, and João P Marques-Silva. Empirical Study of the Anatomy of Modern SAT Solvers. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 343–356. Springer, 2011.

[161] Zurab Khasidashvili and Konstantin Korovin. Predicate Elimination for Preprocessing in First-Order Theorem Proving. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 361–372. Springer, 2016.

[162] Benjamin Kiesl and Martin Suda. A Unifying Principle for Clause Elimination in First-Order Logic. In *International Conference on Automated Deduction*, pages 274–290. Springer, 2017.

[163] Benjamin Kiesl, Martin Suda, Martina Seidl, Hans Tompits, and Armin Biere. Blocked Clauses in First-Order Logic. *arXiv preprint arXiv:1702.00847*, 2017.

[164] Niklas SÃűrensson Koen Claessen. Paradox First-Order Logic Model-Finder. `https://github.com/c-cube/paradox`, accessed 09/2019.

[165] Konstantin Korovin. iProver – An Instantiation-Based Theorem Prover for First-Order Logic (system description). In *International Joint Conference on Automated Reasoning*, pages 292–298. Springer, 2008.

[166] Konstantin Korovin. Inst-Gen – A Modular Approach to Instantiation-Based Automated Reasoning. In *Programming Logics*, pages 239–270. Springer, 2013.

[167] Konstantin Korovin. Non-Cyclic Sorts for First-Order Satisfiability. In Pascal Fontaine, Christophe Ringeissen, and Renate A. Schmidt, editors, *Frontiers of Combining Systems*, pages 214–228. Springer, 2013.

[168] E Kotelnikov. Checking Network Reachability Properties by Automated Reasoning in First-Order Logic. *Automated Theorem Proving with Extensions of First-Order Logic*, pages 114–131.

[169] Evgenii Kotelnikov. *Automated Theorem Proving with Extensions of First-Order Logic.* Department of Computer Science and Engineering, Chalmers University of ?, 2018.

[170] Manolis Koubarakis and Kostis Kyzirakos. Modeling and Querying Metadata in the Semantic Sensor Web: The Model stRDF and the Query Language stSPARQL. In *Extended Semantic Web Conference*, pages 425–439. Springer, 2010.

[171] Laura Kovács and Andrei Voronkov. Finding Loop Invariants for Programs over Arrays using a Theorem Prover. In *International Conference on Fundamental Approaches to Software Engineering*, pages 470–485. Springer, 2009.

[172] Laura Kovács and Andrei Voronkov. First-Order Theorem Proving and Vampire. In *International Conference on Computer Aided Verification*, pages 1–35. Springer, 2013.

[173] Andreas Kuehlmann, Viresh Paruthi, Florian Krohm, and Malay K Ganai. Robust Boolean Reasoning for Equivalence Checking and Functional Property Verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 21(12):1377–1394, 2002.

[174] Werner Kuhn. Metaphors create theories for users. In *European Conference on Spatial Information Theory*, pages 366–376. Springer, 1993.

[175] O Kullmann. A Systematical Approach to 3-SAT Decision, yielding 3-SAT Decision in less than 1: 5045n steps. *Theoretical Computer Science*, 1997.

[176] Oliver Kullmann. The SAT 2005 Solver Competition on Random Instances. *Journal on Satisfiability, Boolean Modeling and Computation*, 2(1-4):61–102, 2006.

[177] Wolfgang Kunz and Dhiraj K Pradhan. Recursive Learning: An Attractive Alternative to the Decision Tree for Test Generation in Digital Ci. In *Proceedings International Test Conference 1992*, page 816. IEEE, 1992.

[178] Kostis Kyzirakos, Manos Karpathiotakis, and Manolis Koubarakis. Developing Registries for the Semantic Sensor Web using stRDF and stSPARQL. In *International Workshop on Semantic Sensor Networks*, 2010.

[179] Shie-Jue Lee and David A Plaisted. Eliminating Duplication with the Hyper-Linking Strategy. *Journal of Automated Reasoning*, 9(1):25–42, 1992.

[180] Reinhold Letz and Gernot Stenz. Proof and Model Generation with Disconnection Tableaux. In *International Conference on Logic for Programming Artificial Intelligence and Reasoning*, pages 142–156. Springer, 2001.

[181] Chu Min Li. Integrating Equivalency Reasoning into Davis-Putnam Procedure. *AAAI/IAAI*, 2000:291–296, 2000.

[182] Chu Min Li and Anbulagan Anbulagan. Heuristics Based on Unit Propagation for Satisfiability Problems. In *Proceedings of the 15th international joint conference on Artifical intelligence-Volume 1*, pages 366–371. Morgan Kaufmann Publishers Inc., 1997.

[183] Chu-Min Li, Fan Xiao, Mao Luo, Felip Manyà, Zhipeng Lü, and Yu Li. Clause Vivification by Unit Propagation in CDCL SAT Solvers. *Artificial Intelligence*, 279:103197, 2020.

[184] Zhiyu Liu, Meng Jiang, and Hai Lin. A Graph-Based Spatial Temporal Logic for Knowledge Representation and Automated Reasoning in Cognitive Robots. *arXiv preprint arXiv:2001.07205*, 2020.

[185] Neha Lodha, Sebastian Ordyniak, and Stefan Szeider. A SAT Approach to Branchwidth. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 179–195. Springer, 2016.

[186] Neha Lodha, Sebastian Ordyniak, and Stefan Szeider. SAT-Encodings for Special Treewidth and Pathwidth. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 429–445. Springer, 2017.

[187] Zhiguo Long, Matt Duckham, Sanjiang Li, and Steven Schockaert. Indexing Large Geographic Datasets with Compact Qualitative Representation. *International Journal of Geographical Information Science*, 30(6):1072–1094, 2016.

[188] Florian Lonsing, Fahiem Bacchus, Armin Biere, Uwe Egly, and Martina Seidl. Enhancing Search-Based QBF Solving by Dynamic Blocked Clause Elimination. In *Logic for Programming, Artificial Intelligence, and Reasoning*, pages 418–433. Springer, 2015.

[189] Feng Lu, Li-C Wang, Kwang-Ting Cheng, and Ric CY Huang. A Circuit SAT Solver with Signal Correlation Guided Learning. In *Proceedings of the conference on Design, Automation and Test in Europe-Volume 1*, page 10892. IEEE Computer Society, 2003.

[190] Peter Lucas. The Representation of Medical Reasoning Models in Resolution-Based Theorem Provers. *Artificial Intelligence in Medicine*, 5(5):395–414, 1993.

[191] I. Lynce and J. Marques-silva. Efficient Data Structures for Fast SAT Solvers. 01 2002.

[192] Inês Lynce and João Marques-Silva. Probing-Based Preprocessing Techniques for Propositional Satisfiability. In *Proceedings. 15th IEEE International Conference on Tools with Artificial Intelligence*, pages 105–110. IEEE, 2003.

[193] Inês Lynce and Joao P Marques-Silva. The Puzzling Role of Simplification in Propositional Satisfiability. In *Proceedings of the EPIA Workshop on Constraint*

Satisfaction and Operational Research Techniques for Problem Solving, pages 73–86, 2001.

[194] Inês Lynce and João P Marques-Silva. An Overview of Backtrack Search Satisfiability Algorithms. *Annals of Mathematics and Artificial Intelligence*, 37(3):307–326, 2003.

[195] Peter Maandag, Henk Barendregt, and Alexandra Silva. *Solving 3-SAT*. PhD thesis, Citeseer, 2012.

[196] Joao Marques-Silva. *Search algorithms for satisfiability problems in combinational switching circuits*. PhD thesis, University of Michigan, 1995.

[197] Joao Marques-Silva. The Impact of Branching Heuristics in Propositional Satisfiability Algorithms. In *Portuguese Conference on Artificial Intelligence*, pages 62–74. Springer, 1999.

[198] Claudio Masolo, Stefano Borgo, Aldo Gangemi, Nicola Guarino, and Alessandro Oltramari. Wonderweb Deliverable D18 - Ontology Library (Final Report). Technical report, National Research Council - Institute of Cognitive Sci. and Technology, Trento, 2003.

[199] William McCune. A Davis-Putnam Program and its Application to Finite First-Order Model Search: Quasigroup Existence Problems. Technical report, Technical report, Argonne National Laboratory, 1994.

[200] William McCune. Mace4 reference manual and guide. *arXiv preprint cs/0310055*, 2003.

[201] Sheila McIlraith and Eyal Amir. Theorem proving with Structured Theories. In *IJCAI*, volume 1, pages 624–634, 2001.

[202] Mark McKenney, Alejandro Pauly, Reasey Praing, and Markus Schneider. Dimension-refined topological predicates. In *Conf. on Advances in Geographic Information Systems (GIS-05)*, pages 240–249. ACM, 2005.

[203] Mark McKenney, Alejandro Pauly, Reasey Praing, and Markus Schneider. Dimension-Refined Topological Pxredicates. In *Proceedings of the 13th annual ACM international workshop on Geographic information systems*, pages 240–249, 2005.

[204] Elliott Mendelson. *Introduction to mathematical logic*. Chapman and Hall/CRC, 2009.

[205] Stephan Mertens, Marc Mézard, and Riccardo Zecchina. Threshold Values of Random K-SAT from the Cavity Method. *Random Structures & Algorithms*, 28(3):340–373, 2006.

[206] David Mitchell, Bart Selman, and Hector Levesque. Hard and Easy Distributions of SAT Problems. In *AAAI*, volume 92, pages 459–465, 1992.

[207] David G Mitchell and Hector J Levesque. Some Pitfalls for Experimenters with Random SAT. *Artificial Intelligence*, 81(1-2):111–125, 1996.

[208] Michael Mitzenmacher. Tight Thresholds for the Pure Literal Rule. *DEC/SRC Technical Note 1997*, 11, 1997.

[209] Matthew W Moskewicz, Conor F Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an Efficient SAT Solver. In *Proceedings of the 38th annual Design Automation Conference*, pages 530–535. ACM, 2001.

[210] DoRon B Motter and Igor L Markov. A Compressed Breadth-First Search for Satisfiability. In *Workshop on Algorithm Engineering and Experimentation*, pages 29–42. Springer, 2002.

[211] Juan Antonio Navarro and Andrei Voronkov. Proof Systems for Effectively Propositional Logic. In *International Joint Conference on Automated Reasoning*, pages 426–440. Springer, 2008.

[212] M Saqib Nawaz, Moin Malik, Yi Li, Meng Sun, and M Lali. A Survey on Theorem Provers in Formal Methods. *arXiv preprint arXiv:1912.03028*, 2019.

[213] Ian Niles and Adam Pease. Towards a Standard Upper Ontology. In *Conf. on Formal Ontology in Inf. Systems (FOIS-01)*, pages 2–9. IOS Press, 2001.

[214] Yakov Novikov. Local Search for Boolean Relations on the Basis of Unit Propagation. In *Proceedings of the conference on Design, Automation and Test in Europe-Volume 1*, page 10810. IEEE Computer Society, 2003.

[215] Eugene Nudelman, Kevin Leyton-Brown, Holger H Hoos, Alex Devkar, and Yoav Shoham. Understanding Random SAT: Beyond the Clauses-to-Variables Ratio. In *International Conference on Principles and Practice of Constraint Programming*, pages 438–452. Springer, 2004.

[216] Richard Ostrowski, Éric Grégoire, Bertrand Mazure, and Lakhdar Sais. Recovering and Exploiting Structural Knowledge from CNF Formulas. In *International Conference on Principles and Practice of Constraint Programming*, pages 185–199. Springer, 2002.

[217] Oded Padon. Deductive Verification of Distributed Protocols in First-Order Logic. In *2018 Formal Methods in Computer Aided Design (FMCAD)*, pages 1–1. IEEE, 2018.

[218] Adam Pease and Geoff Sutcliffe. First order reasoning on a large ontology. *ESARLT*, 257, 2007.

[219] Francis Jeffry Pelletier, Geoff Sutcliffe, and Christian Suttner. The Development of CASC. *AI Communications*, 15(2, 3):79–90, 2002.

[220] Matthew Perry and John Herring. OGC GeoSPARQL – a geographic query language for RDF data, 2012.

[221] David A Plaisted and Yunshan Zhu. Ordered Semantic Hyper-linking. *Journal of Automated Reasoning*, 25(3):167–217, 2000.

[222] Mukul R Prasad, Armin Biere, and Aarti Gupta. A Survey of Recent Advances in SAT-Based Formal Verification. *International Journal on Software Tools for Technology Transfer*, 7(2):156–173, 2005.

[223] Ian Pratt-Hartmann. First-order Mereotopology. In *Handbook of spatial logics*, pages 13–97. Springer, 2007.

[224] Deepak Ramachandran, Pace Reagan, and Keith Goolsbey. First-orderized Research Cyc: Expressivity and Efficiency in a Common-Sense Ontology. In *AAAI Workshop on Contexts and Ontologies: Theory, Practice and Applications*, 2005.

[225] David A. Randell, Anthony G. Cohn, and Zhan Cui. Computing Transivity Tables: A Challenge for Automated Theorem Provers. In *Conf. on Automated Deduction (CADE 1992)*, LNCS 607, pages 786–790. Springer, 1992.

[226] David A. Randell, Zhan Cui, and Anthony G. Cohn. A spatial logic based on regions and connection. In *KR'92: Principles of Knowledge Representation and Reasoning*, pages 165–176, 1992.

[227] David A Randell, Zhan Cui, and Anthony G Cohn. A Spatial Logic Based on Regions and Connection. 1992.

[228] Colin R Reeves and Mériéma Aupetit-Bélaidouni. Estimating the Number of Solutions for SAT Problems. In *International Conference on Parallel Problem Solving from Nature*, pages 101–110. Springer, 2004.

[229] Giles Reger. Some Thoughts About FOL-Translations in Vampire. In *ARQNL@ IJCAR*, pages 11–25, 2018.

[230] Giles Reger and Martin Suda. The Uses of SAT Solvers in Vampire. In *Vampire Workshop*, pages 63–69, 2015.

[231] Andrew Reynolds, Radu Iosif, and Cristina Serban. Reasoning in the Bernays-schönfinkel-ramsey Fragment of Separation Logic. In *International Conference on Verification, Model Checking, and Abstract Interpretation*, pages 462–482. Springer, 2017.

[232] Andrew Reynolds, Cesare Tinelli, Amit Goel, Sava Krstić, Morgan Deters, and Clark Barrett. Quantifier Instantiation Techniques for Finite Model Finding in SMT. In *International Conference on Automated Deduction*, pages 377–391. Springer, 2013.

[233] Alexandre Riazanov and Andrei Voronkov. Splitting without Backtracking. In *IJCAI*, pages 611–617, 2001.

[234] Neil Robertson and Paul D. Seymour. Graph Minors. II. Algorithmic Aspects of Tree-Width. *Journal of algorithms*, 7(3):309–322, 1986.

[235] Lawrence Ryan. *Efficient algorithms for clause-learning SAT solvers.* PhD thesis, Theses (School of Computing Science)/Simon Fraser University, 2004.

[236] Marko Samer and Stefan Szeider. Fixed-Parameter Tractability. *Handbook of Satisfiability*, (part 1), 2009.

[237] Marko Samer and Stefan Szeider. Constraint Satisfaction with Bounded Treewidth Revisited. *Journal of Computer and System Sciences*, 76(2):103–114, 2010.

[238] Markus Schneider and Thomas Behr. Topological Relationships between Complex Spatial Objects. *ACM Trans. Database Systems*, 31(1):39–81, 2006.

[239] Michael Schneider and Geoff Sutcliffe. Reasoning in the OWL2 Full Ontology Language using First-Order Automated Theorem Proving. In *CADE 2011*, pages 461–475. Springer, 2011.

[240] Stephan Schulz. A Comparison of Different Techniques for Grounding Near-Propositional CNF Formulae. In *FLAIRS Conference*, pages 72–76, 2002.

[241] Stephan Schulz, Geoff Sutcliffe, Josef Urban, and Adam Pease. Detecting Inconsistencies in Large First-Order Knowledge Bases. In *International Conference on Automated Deduction*, pages 310–325. Springer, 2017.

[242] Johann Schumann. Automated Theorem Proving in High-Quality Software Design. In *Intellectics and Computational Logic*, pages 295–312. Springer, 2000.

[243] Alexander Scivos and Bernhard Nebel. Double-crossing: Decidability and Computational Complexity of a Qualitative Calculus for Navigation. In *International Conference on Spatial Information Theory*, pages 431–446. Springer, 2001.

[244] R Sekar, IV Ramakrishnan, and Andrei Voronkov. Term Indexing. In *Handbook of automated reasoning*, pages 1853–1964. Elsevier Science Publishers BV, 2001.

[245] Bart Selman, Henry A Kautz, and Bram Cohen. Noise Strategies for Improving Local Search. In *AAAI*, volume 94, pages 337–343, 1994.

[246] Bart Selman and Scott Kirkpatrick. Critical Behavior in the Computational Cost of Satisfiability Testing. *Artificial Intelligence*, 81(1-2):273–295, 1996.

[247] Bart Selman, David G Mitchell, and Hector J Levesque. Generating Hard Satisfiability Problems. *Artificial intelligence*, 81(1-2):17–29, 1996.

[248] Natarajan Shankar. Automated Deduction for Verification. *ACM Computing Surveys (CSUR)*, 41(4):1–56, 2009.

[249] João P Marques Silva and Karem A Sakallah. GRASP – A New Search Algorithm for Satisfiability. In *The Best of ICCAD*, pages 73–89. Springer, 2003.

[250] Iouliia Skliarova and Antonio de Brito Ferrari. Reconfigurable Hardware SAT Solvers: A Survey of Systems. *IEEE Transactions on Computers*, 53(11):1449–1461, 2004.

[251] Barry Smith. Mereotopology: A Theory of Parts and Boundaries. *Data & Knowledge Engineering*, 20(3):287–303, 1996.

[252] Barry Smith et al. *Basic Formal Ontology 2.0 Draft Specification Guide*, 2012.

[253] Barry Smith and Achille C Varzi. The Niche. *Noûs*, 33(2):214–238, 1999.

[254] Niklas Sörensson and Armin Biere. Minimizing Learned Clauses. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 237–243. Springer, 2009.

[255] Christian Steinruecken. SAT-Solving: Performance Analysis of Survey Propagation and DPLL. 2007.

[256] Shirly Stephen and Torsten Hahmann. Formal Qualitative Spatial Augmentation of the Simple Feature Access Model. In *14th International Conference on Spatial Information Theory (COSIT 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[257] Kristin Stock. A Geometric Configuration Ontology to Support Spatial Querying. 2014.

[258] Markus Stocker and Evren Sirin. Pelletspatial: A hybrid rcc-8 and rdf/owl reasoning and query engine. In *OWLED*, volume 529. Citeseer, 2009.

[259] Sathiamoorthy Subbarayan and Dhiraj K Pradhan. NiVER: Non-Increasing Variable Elimination Resolution for Preprocessing SAT Instances. In *International conference on theory and applications of satisfiability testing*, pages 276–291. Springer, 2004.

[260] Geoff Sutcliffe. The TPTP Problem Library and Associated Infrastructure: The FOF and CNF parts, v3.5.0. *J. Automat. Reasoning*, 43(4):337–362, 2009.

[261] Geoff Sutcliffe. The CADE ATP System Competition = CASC. *AI Magazine*, 37(2):99–101, 2016.

[262] Geoff Sutcliffe. The CADE-27 Automated Theorem Proving System Competition–CASC-27. *AI Communications*, 32(5-6):373–389, 2019.

[263] Geoff Sutcliffe, Stephan Schulz, Koen Claessen, and Allen Van Gelder. Using the TPTP language for Writing Derivations and Finite Interpretations. In *International Joint Conference on Automated Reasoning*, pages 67–81. Springer, 2006.

[264] Geoff Sutcliffe and Christian Suttner. The TPTP Problem Library. *Journal of Automated Reasoning*, 21(2):177–203, 1998.

[265] Geoff Sutcliffe and Christian Suttner. The state of CASC. *AI Communications*, 19(1):35–48, 2006.

[266] Geoff Sutcliffe and Christian B. Suttner. The CADE-18 ATP System Competition. *Journal of Automated Reasoning*, 31(1):23–32, 2003.

[267] Stefan Szeider. On ixed-Parameter Tractable Parameterizations of SAT. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 188–202. Springer, 2003.

[268] Stefan Szeider. Not so easy problems for tree decomposable graphs. *arXiv preprint arXiv:1107.1177*, 2011.

[269] Tanel Tammet. Finite Model Muilding: improvements and comparisons. In *CADE-19, Workshop W*, volume 4. Citeseer, 2003.

[270] Josef Urban, Krystof Hoder, and Andrei Voronkov. Evaluation of Automated Theorem Proving on the Mizar Mathematical Library. In *International Congress on Mathematical Software*, pages 155–166. Springer, 2010.

[271] E Lynn Usery and Dalia Varanka. Design and Development of Linked Data from the National Map. *Semantic web*, 3(4):371–384, 2012.

[272] Miroslav N Velev and Randal E Bryant. Effective use of Boolean Satisfiability Procedures in the Formal Verification of Superscalar and VLIW Microprocessors. *Journal of Symbolic Computation*, 35(2):73–106, 2003.

[273] Andrei Voronkov. AVATAR: The architecture for First-Order Theorem Provers. In *International Conference on Computer Aided Verification*, pages 696–710. Springer, 2014.

[274] Michael Wessel and Ralf Möller. Flexible Software Architectures for Ontology-Based Information Systems. *Journal of Applied Logic*, 7(1):75–99, 2009.

[275] Siert Wieringa and Keijo Heljanko. Concurrent Clause Strengthening. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 116–132. Springer, 2013.

[276] Johan Wittocx, Maarten Mariën, and Marc Denecker. Grounding FO and FO (ID) with Bounds. *Journal of Artificial Intelligence Research*, 38:223–269, 2010.

[277] Andreas Wotzlaw, Alexander van der Grinten, and Ewald Speckenmeyer. Effectiveness of pre-and inprocessing for CDCL-based SAT solving. *arXiv preprint arXiv:1310.4756*, 2013.

[278] Emmanuel Zarpas. Benchmarking SAT Solvers for Bounded Model Checking. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 340–354. Springer, 2005.

[279] Hantao Zhang and Jian Zhang. MACE4 and SEM: A Comparison of Finite Model Generators. In Maria Paola Bonacina and Mark E. Stickel, editors, *Automated Reasoning and Mathematics: Essays in Memory of William W. McCune*, pages 101–130. Springer, 2013.

[280] Jian Zhang. Constructing Finite Algebras with FALCON. *Journal of automated reasoning*, 17(1):1–22, 1996.

[281] Jian Zhang and Hantao Zhang. SEM: A System for Enumerating Models. In *IJCAI*, volume 95, pages 298–303, 1995.

[282] Lintao Zhang. On Subsumption Removal and on-the-fly CNF Simplification. In *International conference on theory and applications of satisfiability testing*, pages 482–489. Springer, 2005.

APPENDIX A

SUPPLEMENTARY MATERIAL

SF-FOL	**CODI/CODIB**	**RCC**	**INCH**
sf_point	Point	-	-
sf_curve	Curve	-	–
sf_surface	ArealRegion	-	
within	Cont	PP/NTTP	INCH/CS/CH
crosses	Inc	-	-
overlaps	PO	O	OV
intersects	C	P	-
touches	SC	EC	-

Table A.1: Mapping between SF-FOL terms and concepts in CODIB, RCC and INCH ontologies. We use this mapping to construct sample (r-d)ABoxes for each theory from the Master ABox.

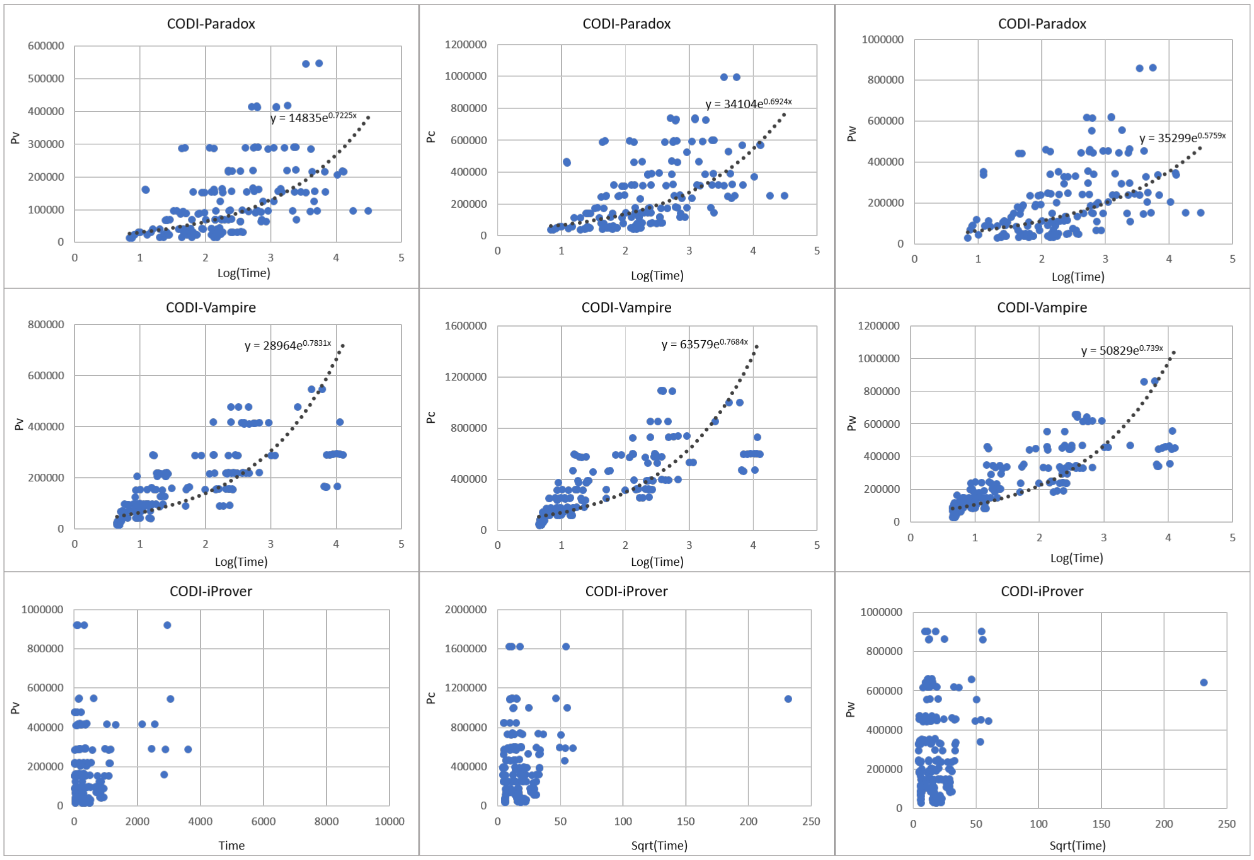


Figure A.1: Dependencies between model finding time and three measures of size of the SAT problem: no. of propositional variables, no. of propositional clauses and approximated number of those clauses having three or more literals.

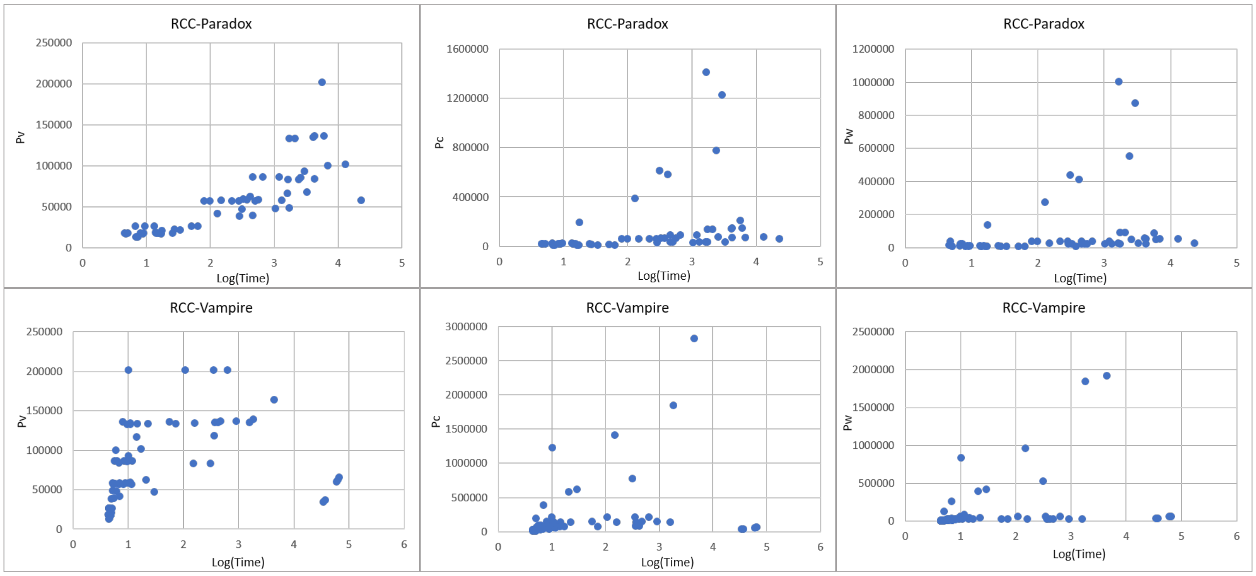


Figure A.2: Dependencies between model finding time of RCC and size measures of the SAT problem.

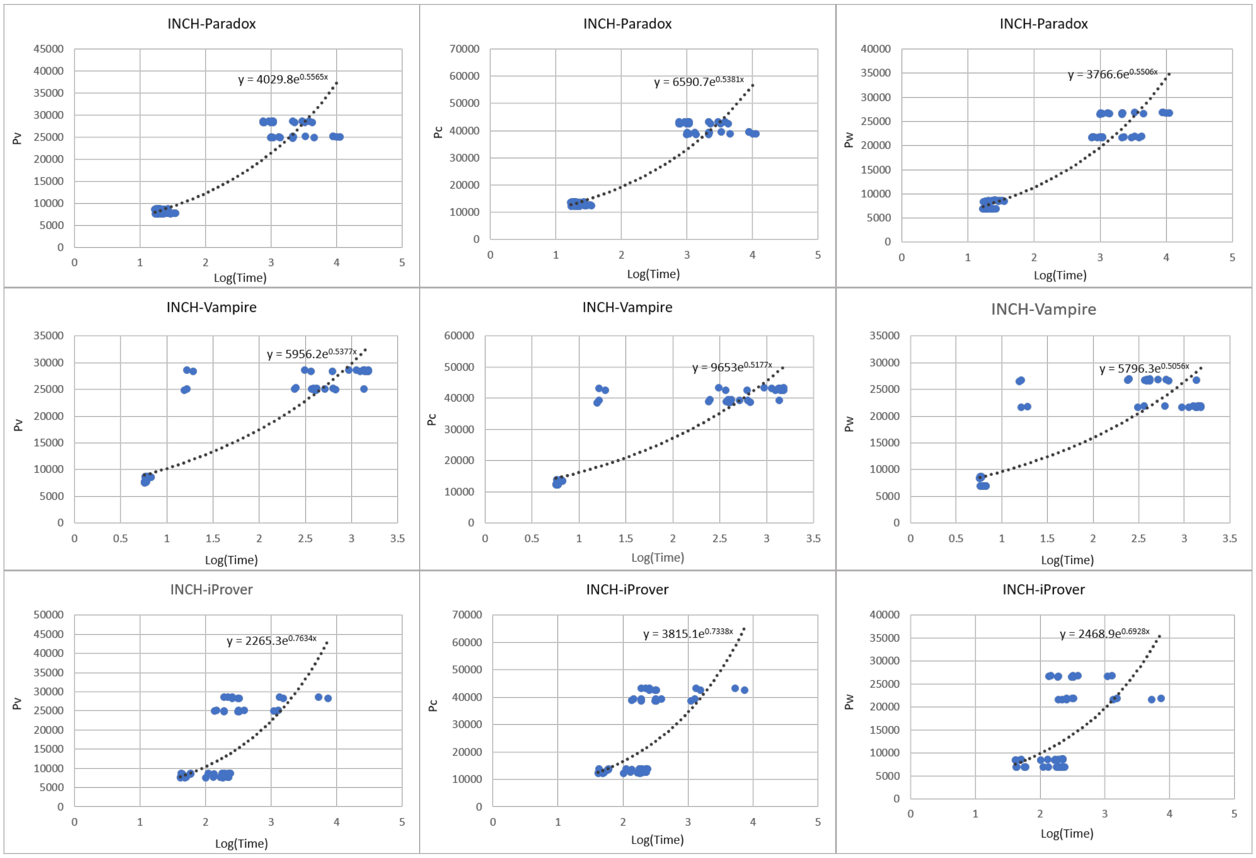


Figure A.3: Dependencies between model finding time of INCH and size measures of the SAT problem.

BIOGRAPHY OF THE AUTHOR

Shirly Stephen is a candidate for the Doctor of Philosophy degree in Spatial Information Science and Engineering from the University of Maine in December 2021.