Using a SAT-based Model Finder to Verify First-Order Logic Ontologies of Space against Datasets: Scalability and Bottlenecks in Practice

Shirly Stephen and Torsten Hahmann
School of Computing and Information Science
University of Maine, Orono, ME 04469

Abstract
Semantic verification of an ontology encompasses checking its internal consistency, which ensures the ontology itself is free of contradictions, and its external consistency, which ensures that it is consistent with the kind of datasets it is intended to work with. Unlike description logic (DL) ontologies, first-order logic (FOL) ontologies are rarely verified externally against datasets because FOL model finders have been limited to tiny models with less than 20 individuals.

Focusing on FOL ontologies of spatial relations, this paper investigates the source of the bottleneck via a formal analysis and experiments with the SAT-based model finder Paradox. We demonstrate that the presence of many defined terms of highest arity (here defined binary spatial relations), which are common in ontologies but not in other mathematical theories, significantly slows down model finding. It is shown that removing optional explicit definitions and substituting the terms they define with their definiens exponentially speeds up model finding, allowing easy construction of models with more than 100 individuals. While this is still small compared to what DL reasoners can handle, it is a significant improvement over the tiny, often trivial models to which verification of FOL ontologies has been traditionally restricted.

1 Introduction
Formal ontologies capture a domain’s terminology and its semantics in a logic-based language as a means to automatically reason about the domain. Such ontologies may provide the background knowledge necessary to interpret a dataset collected in the domain, to semantically integrate different datasets or applications, or to make implicit assumptions in the domain explicitly provable. However, an ontology can only serve its purpose if we know that it correctly and adequately captures the modeled domain. Ontology evaluation comprises techniques and measures of ontologies’ correctness and adequacy, wherein ontology verification aims to verify an ontology’s correctness (Vrandečić 2009; Gómez-Pérez 2004) and ontology validation its adequacy (Obrst et al. 2007; Gómez-Pérez 2004). While ontology validation requires significant human intervention, many aspects of ontology verification benefit from extensive automation (Grüninger et al. 2010), including checking an ontology’s logical consistency. This comes in two forms: (1) checking an ontology’s internal consistency that rules out contradicting states by generating some model, and (2) checking its external consistency with datasets that are representative of the ontology’s intended domain or application.

While description logic (DL) ontologies – including OWL ontologies – can be efficiently verified internally and externally even with large datasets (i.e., a large ABox), first-order logic (FOL) ontologies are often only internally verified. One reason for this is that FOL ontologies often exclusively formalize the structure of domain (i.e., the terms and how they can be interpreted) and rarely contain facts/data points about individuals (i.e., they typically lack an ABox). But more importantly, model finding for FOL ontologies is not only theoretically incomplete but has also not been very successful in practice either except for tiny, often trivial models. This paper examines this assumption more closely and identifies specific bottlenecks that can be remedied for model finding to scale better in practice despite its theoretical undecidability and intractability.

Model finding for FOL ontologies typically utilizes FOL model finders which are often lumped together with other automated theorem provers (ATP) but which focus on proving satisfiability rather than unsatisfiability of a logical theory. However, the available model finders, such as the SAT-based model finders Paradox (Claessen and Sörensson 2003) or Mace4 (McCune 2003), are mostly tested on relatively small axiomatizations with few nonlogical symbols (i.e., predicate and function symbols), as commonly found in mathematical conjectures. To search for models, SAT-based model finders convert a FOL theory into Clausal Normal Form (CNF) and then instantiate it with (an increasing number of) individuals to produce a series of propositional satisfiability (SAT) problems, whose size (as measured in number of propositional variables and clauses) grows exponentially with the number of individuals in a model and the size (number and arity of predicates) of the ontology’s terminology.

Objectives This paper examines to what extent model finding for FOL ontologies, with and without data, is feasible in practice. Our theoretical analysis reveals that a major source of intractability is the number and arity of terms. But unlike other mathematical theories, many terms are ex-
licitly defined in ontologies and can be substituted by their definiens. This allows us to directly measure the impact additional defined terms have on reasoning efficiency and scalability in order to test three hypotheses:

1. Additional explicit definitions limits the size of models that can be constructed and negatively impacts the time to find a model, regardless of whether the defined terms are actually used in the ABox.

2. Rewriting ABox facts that use defined terms with their definiens speeds and scales up model finding.

3. More complex definitions slow down model finding more.

**Approach** We test our hypotheses specifically on FOL spatial ontologies and the SAT-based model finder Paradox, which in almost all our preliminary experiments performed better than Mace4. We study how additional definitions of varying complexity impact model finding as compared to logically equivalent ontologies without these definitions. We further test whether rewriting assertions (i.e., facts) about individuals using the definiens rather than a defined term improves model finding performance. We conduct these experiments with ABoxes of increasing domain sizes (as determined by the number of distinct individuals) and with definitions of varying complexity, identifying up to what domain sizes we can find models and measuring the time needed to find one. We first explore how model finding with FOL ontologies generally scales before we examine how the specific factors from our hypotheses (number and complexity of definitions, use of defined terms in the ABox) influence model finding performance.

From our experience, there is a lot of unexplained variability in the performance of automated reasoning with FOL ontologies that arises from seemingly minor differences in ontologies. Thus, our experiments aim to reduce differences in other factors by working with a set of logically equivalent ontologies to keep the number of possible models constant regardless of whether extra definitions are present or not.

**Contributions** While our experiments are limited to one family of spatial ontologies, they suggest that the elimination of optional definitions and corresponding rewriting of ABoxes is surprisingly effective in making model finding for FOL ontologies scale better. Thus, this work shows that external verification of complex midsize FOL ontologies with many defined terms but a small number of primitives is feasible in practice, though it will likely never scale like external verification of DL ontologies.

Another important task that requires satisfiability proving for ontologies with ABoxes involves the identification of ontologies (e.g., spatial ontologies) that are suitable (i.e., consistent) background theories for a given dataset. This can be accomplished by checking the consistency of different candidate ontologies with samples of the given dataset (i.e. a set of spatially grounded facts). This not only helps overcome the hurdle of selecting ontologies for reuse, but can also be used for automated commonsense reasoning that fills in background knowledge to answer queries more intelligently. This scenario is of growing importance because for many domains there are multiple competing ontologies of different strength as, for example, collected in the ontology repository COLORE.

Spatial knowledge, in particular, this identification of suitable background theories is only possible with spatial ontologies in very expressive KR languages, as only a small portion of such spatial background knowledge is captured by qualitative spatial calculi (in the RCC (Randell, Cui, and Cohn 1992) or more powerful calculi such as presented in (Cohn et al. 2014)) or in less expressive DLs (e.g. (Katz and Grau 2005; Stocker and Sirin 2009)).

**Structure** Sec. 2 reviews the theoretical foundations, Sec. 3 discusses related work and Sec. 4 lays the formal foundation for working with facts and definitions in FOL ontologies. Section 5 present our experiments and results.

## 2 Preliminaries

This section briefly reviews the basics of FOL ontologies and the terms TBox, RBox and ABox from Description Logics. This section introduces the theoretical background for our work. We first review first-order logic and introduce definitions for the TBox and ABox of a FOL ontology. Subsequently, we review the different kinds of reasoning tasks typically performed on ontologies and then specifically review SAT-based model finding as a specific approach to proving consistency of an FOL ontology. As much as possible, we stick with standard terminology from first-order logic and automated reasoning.

### 2.1 First-order logic (FOL)

A first-order logic (FOL) ontology $T$ is a set of FOL sentences. The nonlogical symbols, i.e., all constants, function symbols, and predicates, mentioned in $T$ form its signature, denoted by $\lambda(T)$. For simplicity, we restrict the signature to constants and predicates only, because each function symbol can be represented using a predicate symbol. A FOL sentence is a FOL formula wherein no variables appear free. Formulas are constructed from atoms using the logical connectives $\land, \lor, \rightarrow, \leftrightarrow$ and $\neg$ and the quantifiers $\forall$ and $\exists$. An atom is of the form $P(t_1, ..., t_n)$ where $P$ is a predicate symbol in $\lambda(T)$ of arity $n$ and all $t_i$ are terms. Since we restrict ourselves to function-free signatures, terms are either constants or variables and ground atoms are atoms without variables, that is, with constants as only terms. A dataset is assumed to contain only ground atoms, which we refer to as facts or assertions about individuals.

Each FOL ontology $T$ admits a set of interpretations over a nonempty domain $D$ of individuals. Any interpretation assigns each constant in $\lambda(T)$ an individual in $D$ and each n-ary predicate $\Omega$ in $\lambda(T)$ a relation $\Psi(\Omega) : D^n \rightarrow \{\text{True}, \text{False}\}$. An interpretation $I$ for which all sentences in

---

1Intuitively, the number of models greatly influences model finding performance as more models result in a higher chance to encounter a model during the SAT solving process and thus faster runtime on average.

2http://colore.oor.net
which are positive or negated atoms, i.e. either domain sizes. This paper does not intend to compare model and associated memory consumption issues experienced in logic instead of propositional logic. These avert the size on a conversion to a more expressive function-free clause (Baumgartner et al. 2009) use specialized calculi that operating a finite model prove satisfiability (consistency) of an ontology by generating satisfiability, proving entailments (including unsatisfiability) of an ontology or, in a similar fashion, prove theorems about an ontology, and model finders, which aim to as automated theorem provers (ATPs) – typically support one or more of three fundamental reasoning tasks: proving satisfiability, proving entailments (including unsatisfiability), and answering queries. ATPs fall into two categories: theorem provers, which can prove unsatisfiability (inconsistency) of an ontology or, in a similar fashion, prove theorems about an ontology, and model finders, which aim to prove satisfiability (consistency) of an ontology by generating a finite model.\footnote{This only works for ontologies that admit some finite models.} In the presented study, we are concerned with model finding only. Far fewer model finders exist than theorem provers (Zhang and Zhang 2013), as also evident from CADE’s automated theorem proving competition (CASC). Some are optimized for mathematical theories that rely heavily on equality and function symbols rather than predicates. FOL model finders typically convert the ontology for increasing domain sizes to a decidable logic. Paradox (Claessen and Sörensson 2003) and Mace4\footnote{Mace4 propositionalizes the problem but applies a more specialized constraint satisfaction algorithm.} convert the problems to propositional logic, essentially creating a series of SAT problems of increasing size until a model is found. Others, such as iProver (Korovin 2013) and Darwin-FM (Baumgartner et al. 2009) 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 a conversion to propositional logic and are claimed to significantly scale better for higher-arity predicates and for larger domain sizes. This paper does not intend to compare model finders against each other, but rather to get a better sense of general bottlenecks and scalability of model construction for FOL ontologies. Thus, this paper exclusively focuses on Paradox as we have had fairly consistent success with it in the verification of FOL ontologies (see, e.g., (Hahmann 2013)). In the future, we plan to compare it to results with iProver, which may lead to improvements in scalability beyond what we report here.

SAT-based model finders convert a FOL ontology to a propositional representation and then treat model finding as showing satisfiability of a propositional satisfiability (SAT) problem. Propositionalization requires instantiating all FOL-CNF clauses, which in turn requires fixing the domain size, i.e. choosing a fixed set of constants over which the variables range in order to instantiate all predicates by propositional variables (Zhang and Zhang 2013). If the domain size (i.e., the number of individuals in a model) is not known in advance, the model finder starts by instantiating the smallest domain size and increases it by one individual each time the search space is exhausted. If, for example, the smallest model has 8 individuals, then the model finder will perform 8 SAT instances: the first 7 are unsatisfiable and the 8th one is satisfiable. The size of a propositional representation of a FOL is measured in terms of number of propositional clauses, variables, and literals per clause. Each FOL-CNF clause leads to an exponentially growing number of propositional clauses, where \(d\) is the domain size and \(v\) is the number of variables in the FOL-CNF clause, because every variables can be independently instantiated with any of the \(d\) individuals. For example, case 1 in our experiments contains 3 FOL-CNF clauses with 3 variables, 32 with 2 and 30 with 1, resulting in \(3 \cdot 40^3 + 32 \cdot 40^2 + 30 \cdot 40 = 244,400\) propositional clauses (each FOL-CNF clause with 3 variables contributing 64,000 propositional clauses, more than all 32 FOL-CNF clauses with 2 variables together). For domain size 80, this increases to 1,743,200 clauses. The same ontology with C as an additional defined binary predicate (case 3) adds only 3 FOL-CNF clauses (less than 5\% increase), but because one clause contains 3 variables, the number of propositional clauses increases to 311,600 and 2,268,800 (an over 27\% increase), respectively, for domain sizes 40 and 80.

The number of propositional variables in the SAT representation is dependent upon the number and arity of predicates: each predicate of arity \(n\) results in \(d^n\) propositional variables where \(d\) is again the domain size. For our example ontology from case 1 with 8 binary and 8 unary predicates, this amounts to \(8 \cdot 40^2 + 8 \cdot 40 = 13,120\) and 51,840 variables for domain sizes 40 and 80. When adding the defined binary predicate C (case 3), this increases to 14,720 and 58,240 for domain sizes 40 and 80. This increase is linear relative to the number of predicates of highest arity but an exponential increase relative to the domain size.

The number of propositional variables determines the search space, which consists of \(2^{3v}\) possible interpretations. For ontologies with only unary and binary predicates – as used in our study – this amounts to \((2^3)(d^2) \cdot (2^v)^d\) possible interpretations, which is exponential in both the number of binary predicates and the domain size. Modern SAT solvers
employ effective strategies, in particular clause learning, to drastically prune the search space and are, thus, able to deal with thousands of variables and tens of thousands of clauses (Gomes et al. 2008). However, the search space can, in theory, be greatly reduced if the number of predicates of highest arity can be kept low. This is the idea tested here experimentally, leveraging the fact that many ontologies have lots of defined predicates, which we can easily dispense of during model finding and which can be added at the end.

3 Related Work

Prior work on better understanding the practical feasibility of automated reasoning has mostly dealt with description logics (DL), a family of knowledge representation languages that are decidable fragments of FOL. For sufficiently restricted DLs, certain reasoning tasks become even tractable. In most DLs, the various reasoning tasks, such as subsumption of concepts, query answering, but also proving satisfiability, can be reduced to satisfiability of concepts (Horrocks, Sattler, and Tobies 2000), thus allowing reasoners to rely on a single general-purpose reasoning procedure such as tableaux. Thus, proving satisfiability (i.e., model finding) even with large ABoxes is fundamentally no different than theorem proving (Haarslev and Möller 2000; Horrocks, Sattler, and Tobies 2000) and scales well, especially for more restricted DLs without role assertions (i.e. with only unary predicates), even for large ABoxes with 100,000 facts or more (De Giacomo and Lenzerini 1996; Horrocks et al. 2004; Motik and Sattler 2006).

Theorem Proving and Query Answering with FOL Ontologies For FOL ontologies, satisfiability proving (i.e., model finding) is fundamentally no different from unsatisfiability proving (i.e., entailment checking/theorem proving, but also used for query answering), with prior studies of the practical scalability of reasoning focused mostly on methods for unsatisfiability proving. Previous work using FOL automated reasoners has demonstrated that query answering and entailment problems can be solved efficiently even for large axiomatic ontologies, such as OWL ontologies translated to FOL (Pease and Sutcliffe 2007; Horrocks and Voronkov 2006; Ramachandran, Reagan, and Goolsbey 2005), though the ABox was usually empty in these studies. For example, Vampire could answer many queries within less than a second for the KIF (a FOL syntax) version of the Suggested Upper Merged Ontology (SUMO) that contains about 5,000 axioms (Pease et al. 2010). However, test results for proving satisfiability were mostly disappointing with many cases never terminating (Tsarkov and Horrocks 2003).

Consistency Checking with FOL Ontologies Existing work in FOL ontology consistency checking is limited to satisfiability verification of terminological axioms or entailment checking (Pease and Sutcliffe 2007; Pease et al. 2010; Ramachandran, Reagan, and Goolsbey 2005), with theorem proving being much more successful at identifying any inconsistencies as compared to proving an ontology’s consistency. For example (Schneider and Sutcliffe 2011) successfully employed first-order theorem provers to reason over ontologies translated from OWL-DL and OWL-Full, but model finding was much less successful.

Comparison of Reasoners Our work is complementary to related work that focuses on comparing the performance of model finders, e.g. (Baumgartner et al. 2009; Korovin 2013; Zhang and Zhang 2013), and of other kinds of reasoners (Sutcliffe and Suttner 2001) which is different from our approach that attempts to identify bottlenecks that specifically arise from the structure of ontologies. While various benchmarks are available for comparing ABox reasoning across DL reasoners as, e.g., reviewed in (Weithöner et al. 2007), we are unaware of existing benchmarks for comparing satisfiability checking of FOL ontologies with nonempty ABoxes. For example the benchmarks in (Pease et al. 2010) do not contain many facts. (Sutcliffe and Suttner 2001) empirically evaluated general purpose ATPs to determine which systems work well for what type of problems. But this evaluation involves few ontologies and none with nonempty ABoxes.

4 Formalization: ABox, TBox and Definitions in FOL Ontologies

DL ontologies can be divided into a TBox, an ABox (De Giacomo and Lenzerini 1996; Horrocks, Sattler, and Tobies 2000), and, for expressive DLs, an RBox. 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). The ABox captures assertions about individuals, i.e., statements about an individual being an instance of a particular concept or being related to another individual via a particular role. We adapt this distinction for FOL ontologies and introduce and formalize the notion of optional explicit definitions to formalize the ideas central to our experiments.

4.1 ABox

A FOL ontology can mix structural knowledge and assertions about individuals, even in a single sentence. Thus, we define its ABox in terms of the ground formulas that result from conversion to FOL-CNF:

Definition 1. Let $T$ be an ontology with signature $\lambda(T)$ and let $C$ be the corresponding set of FOL-CNF clauses. Then $ABox(T)$ is the subset of $T$’s sentences that yield ground clauses in $C$ that only use symbols from $\lambda(T)^5$.

While an ABox may contain disjunctive knowledge – reflected in ground clauses with multiple literals – clauses typically contain only a single literal. Such facts are the only kind of ABox assertions we consider in our experiments. For conceptual simplicity, we further require that the ABox itself is represented as a set of ground clauses.

Definition 2. An ontology $T$’s ABox is clean iff $ABox(T)$ contains only ground clauses with exactly one literal.

5Sentences that result in clauses with new constants or function symbols introduced via skolemization are not part of the ABox.
The ontologies we experiment with contain, like many ontologies, only unary and binary predicates. In such cases, the ABox consists of three types of assertions:

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

**Relational Assertions**, e.g. Inc(i95, penobscotCounty), assert two individuals to be in a certain relation.

**Distinctness Assertions** assure that distinct constants denote distinct individuals. They are automatically added to all our ABoxes, for example stating i95 ≠ i295.

### 4.2 TBox

A FOL ontology’s TBox captures its structural, i.e., non-factual knowledge. While it contains a subset of the ontology’s sentences, we define it indirectly via the set of generated FOL-CNF clauses.

**Definition 3.** Let $T$ be an ontology and let $C$ be the corresponding set of FOL-CNF clauses. Then $TBox(T)$ is the subset of $T$’s sentences that yield clauses in $C \setminus ABox(T)$.

For an ontology with a clean ABox, as is the case in our experiments, we have $TBox(T) = T \setminus ABox(T)$. In this case, the ontology does not mix factual and structural knowledge in a single sentence.

### 4.3 Definitions

We further distinguish explicit definitions of predicates as a special type of TBox sentence:

**Definition 4.** Let $T$ be an ontology with signature $\lambda(T)$. Then an explicit definition of an $n$-ary predicate $\Omega \in \lambda(T)$ in $T$ is a sentence $\sigma \in TBox(T)$ of the form $\forall x_1, \ldots, x_n[\Omega(x_1, \ldots, x_n) \leftrightarrow \alpha(x_1, \ldots, x_n)]$ where $\alpha$ 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 $\Omega$ is said to be **explicitly defined** in $T$.

Optional definitions are explicit definitions of predicates not used in other sentences:

**Definition 5.** Let $T$ be an ontology with signature $\lambda(T)$. An explicit definition $\sigma \in TBox(T)$ of a symbol $\Omega \in \lambda(T)$ is an optional **definition** in $T$ iff $\Omega$ does not appear in any sentence in $TBox(T) \setminus \sigma$.

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

**Definition 6.** Let $T$ be an ontology with signature $\lambda(T)$. Then a **definition set** of $T$ is defined recursively as follows:

**B.** The set of all optional definitions in $TBox(T)$ forms a definition set;

**R.** If $D$ is a definition set of $T$ and $\sigma$ is an optional definition of $\Omega$ in $TBox(T) \setminus D$ then $D' = \{\sigma' | \sigma \in D$ and $\sigma' = \sigma[\Omega(x_1, \ldots, x_n)/\alpha(x_1, \ldots, x_n)] \} \cup \sigma$, that is, $D'$ is replacing all occurrences of $\Omega$ in $D$ by its definition from $\sigma$ and adding $\sigma$ as a new definition.

**Definition 7.** Let $T$ be an ontology with signature $\lambda(T)$. Then $DBox(T)$ is a definition set of $T$ such that no optional definition exists in $TBox(T) \setminus DBox(T)$.

$\Omega \in \lambda(T)$ is **optionally defined in $T$** iff $\Omega$ does not appear in $TBox(T) \setminus DBox(T)$.

To study the impact of optionally defined predicates on model finding performance, we want to remove them from the ABox without changing the ontology’s semantics. This is achieved by replacing sentences that use optionally defined predicates by **defined assertions**.

**Definition 8.** Let $T$ be an ontology and $D$ be a definition set therein. Then $ABox_D(T) = ABox(T) \cup \{\Omega(x_1, \ldots, x_n)/\alpha(x_1, \ldots, x_n) \mid \exists \sigma \in D [\Omega(x_1, x_2, \ldots, x_n) \leftrightarrow \alpha(x_1, x_2, \ldots, x_n)]\}$.

Any sentence $\sigma \in ABox_D(T)$ such that $\sigma \notin ABox(T)$ is called a **defined assertion**.

In other words, $ABox_D(T)$ is $T$’s ABox with all occurrences of predicates $\Omega_i$ that are defined by some definition in $D$ substituted by their definitions $\alpha_i$. Note that an ABox with defined assertions may no longer contain single-literal ground clauses with symbols from $\lambda(T)$. Defined assertions may contain variables introduced during the substitution. For example, a fact $P(\text{‘195’, ‘295w’})$ may result in the defined assertion $\exists x[P(x, \text{‘i95’}) \land P(x, \text{‘295w’})]$ if PO’s definition from Table 1 is part of the definition set.

We can now formally express that removing a definition set from an ontology and substituting all its defined terms does not affect the presence or number of models.

**Theorem 1.** Let $T$ be an ontology and $D$ be a definition set of $T$. Then there is a bijection between the models of $(TBox(T) \setminus D) \cup ABox_D(T)$ and the models of $TBox(T) \cup ABox(T)$, that is, every model of $(TBox(T) \setminus D) \cup ABox_D(T)$ can be uniquely expanded into a model of $TBox(T) \cup ABox(T)$.

**Proof.** Note that from the construction of $ABox_D(T)$ in Def. 8, $D \cup ABox_D(T) \equiv ABox(T)$. Further note that $D$ explicitly defines the set of symbols in $\lambda(T)$ but not used in $(TBox(T) \setminus D) \cup ABox_D(T)$. Then $(TBox(T) \setminus D) \cup ABox_D(T) \equiv TBox(T) \cup ABox(T)$.

We can then apply Beth’s definability theorem (Beth 1953), 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 $(TBox(T) \setminus D) \cup ABox_D(T)$, they are implicitly definable in its models, which become models of $TBox(T) \cup ABox(T)$ by the logical equivalence of the two theories. \[\square\]

Such a bijection exists specifically where $D = DBox(T)$, which captures the maximal set of optional definitions that can be easily removed without altering the ontology’s semantics. This idea forms the basis of our strategy for improving model finding because $(TBox(T) \setminus DBox(T))$ has a smaller signature for which, in theory, model finding should be faster and scale better.

### 5 Experiments and Results

This section describes the three experiments and analyzes their results. But first, we summarize the ontologies and the dataset used for all experiments.
We work with five optionally defined spatial predicates

\[ PP \] (proper parthood), \( C \) (contact: sharing a contained entity), \( PO \) (sharing a part), \( Inc \) (incidence), and \( SC \) (superficial contact), with their full definitions provided in Table 1.

### ABoxes: Spatial Datasets

The map of critical habitat for lynx in Maine\(^6\) forms the master ABox from which samples for all experiments are drawn. It consists of 173 spatial objects, each either a Point (27), Curve (90), or ArealRegion (56), and 3,665 spatial relation assertions (245 positive ones and 3,420 negated ones). Table 2 gives a precise breakdown.

Each experiment is averaged over 10 or more repetitions with different samples from the master ABox. Samples are constructed via a Python script that randomly picks \( n \) concept assertions, where \( n \) is the sample size, then gathers all relational assertions that mention only the selected individuals, and then adds distinctness assertions between the \( n \) individuals. The scripts further substitute facts by defined assertions as described in Experiment 1. An ABox is expected to have on average 3665 · \( (\frac{1}{100})^2 \) relational assertions, which amounts to 196 to 784 for sample sizes 40 to 80. Once concept and distinctness assertions are added, a total of 1,016 to 4,024 facts are expected for sample sizes 40 to 80.

### Computing Infrastructure

All experiments are run on an Intel Xeon CPU E5-2620 v3 at 2.40 GHz (with 12 cores, though a single instance of Paradox does not use more than a single core) with 64GB RAM and 64bit Windows 10 Pro, using a Windows binary of Paradox3 from 07/29/2008. Measured times are CPU time for the process that runs Paradox.

### 5.2 Exp. 1: Scalability of Model Finding with Different Definition Sets

To test our first hypothesis about the impact of optional definitions vs. defined assertions on model-finding, the 13 different cases with different definition sets shown in Table 3 are run with sample sizes for the ABox (i.e., domain size of sought models) ranging from 40 to 80. For each sample size, the experiment is repeated 10 times. To ensure comparability of the results, for any specific sample size, the same 10 ABox samples are used for all 13 cases. The ABoxes only differ in that depending on the case’s definition set, some facts are replaced by the corresponding defined assertions. That is, any of the 5 optional definitions not used in a specific test (i.e., the definition is not included in the TBox for that test) are substituted by their definitions, thus creating 13 variants of the same ABox (i.e., ABox\( \{PP, C, PO, Inc, SC\} \) for case 1, ABox\( \{C, PO, Inc, SC\} \) for case 2, \( \ldots \), ABox\( \{SC\} \) for case 10, etc.) for the 13 cases. For example, all relational assertions in ABox\( \{PP, C, PO, Inc, SC\} \) for case 1 have all facts that use any of PP, C, PO, Inc, or SC rewritten in terms of the definitions’ respective definitions, whereas facts using PP are

---

Table 3: Summary of the TBoxes and the FOL-CNF version of the 13 cases experimented with. Each row represents one case, indicating the included optional definitions, statistics of the resulting FOL-CNF, and example statistics of the resulting propositionalized versions for samples sizes 40, 60, and 80.

The abbreviations denote: $P_b, P_u$: binary and unary predicates; $C$: FOL-CNF clauses; $V$: variables in a FOL-CNF clause; $C_{S,3}$: FOL-CNF clauses with more than 3 literals; $S_{c}, S_{f}$ - Skolem constants and functions introduced in the conversion to FOL-CNF; $P_v, P_i$: propositional variables and clauses; $I$: sample size (i.e., distinct individuals in the ABox samples).

<table>
<thead>
<tr>
<th>TBox Case</th>
<th>Optional definitions</th>
<th>FOL - CNF</th>
<th>$#C$</th>
<th>$#C_{S}$</th>
<th>$#C_{S,3}$</th>
<th>Propositional CNF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$#P_b, #P_u$</td>
<td>$#C_{v=1}$</td>
<td>$#C_{v=2}$</td>
<td>$#C_{v=3}$</td>
<td>$I = 40$</td>
<td>$I = 60$</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>65</td>
<td>3</td>
<td>32</td>
<td>30</td>
<td>31</td>
</tr>
<tr>
<td>2</td>
<td>$\sqrt{\cdot}$</td>
<td>9882</td>
<td>4</td>
<td>15</td>
<td>2294</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>$\sqrt{\cdot}$</td>
<td>18522</td>
<td>13120</td>
<td>2287.2</td>
<td>5 + 3</td>
<td>5 + 2</td>
</tr>
<tr>
<td>4</td>
<td>$\sqrt{\cdot}$</td>
<td>58240</td>
<td>71040</td>
<td>16400</td>
<td>68</td>
<td>$\sqrt{\cdot}$</td>
</tr>
<tr>
<td>5</td>
<td>$\sqrt{\cdot}$</td>
<td>14720</td>
<td>51840</td>
<td>16400</td>
<td>2844.0</td>
<td>5 + 3</td>
</tr>
<tr>
<td>6</td>
<td>$\sqrt{\cdot}$</td>
<td>36480</td>
<td>92240</td>
<td>43680</td>
<td>68</td>
<td>92240</td>
</tr>
<tr>
<td>7</td>
<td>$\sqrt{\cdot}$</td>
<td>3116</td>
<td>5</td>
<td>39</td>
<td>30</td>
<td>35</td>
</tr>
<tr>
<td>8</td>
<td>$\sqrt{\cdot}$</td>
<td>71040</td>
<td>92240</td>
<td>43680</td>
<td>68</td>
<td>92240</td>
</tr>
<tr>
<td>9</td>
<td>$\sqrt{\cdot}$</td>
<td>32880</td>
<td>58240</td>
<td>77440</td>
<td>58240</td>
<td>5 + 3</td>
</tr>
<tr>
<td>10</td>
<td>$\sqrt{\cdot}$</td>
<td>36480</td>
<td>92240</td>
<td>43680</td>
<td>68</td>
<td>92240</td>
</tr>
<tr>
<td>11</td>
<td>$\sqrt{\cdot}$</td>
<td>3116</td>
<td>5</td>
<td>39</td>
<td>30</td>
<td>35</td>
</tr>
<tr>
<td>12</td>
<td>$\sqrt{\cdot}$</td>
<td>71040</td>
<td>92240</td>
<td>43680</td>
<td>68</td>
<td>92240</td>
</tr>
<tr>
<td>13</td>
<td>$\sqrt{\cdot}$</td>
<td>3116</td>
<td>5</td>
<td>39</td>
<td>30</td>
<td>35</td>
</tr>
</tbody>
</table>

Figure 1: Average medial time to find a model (i.e., the average after dropping the lowest and highest 2 times from 10 samples) for the 13 cases detailed in Table 3 and for sample sizes 40 to 80. Any cases not plotted did not terminate successfully.

not rewritten in case 2. Hence the 13 cases (TBox + ABox) are semantically equivalent, that is they have the same number of models, but are syntactically different.

The results of our experiments are plotted in Fig. 1 using averages of the 6 medial times (i.e., the two highest and lowest times are dropped) for each combination of case and sample size, rounded to 3 significant digits. Notice that the figure uses a logarithmic scale. We use a timeout of 100,000 seconds in all experiments, but never actually reach that limit. Instead, cases where no model is found (not shown in the figure) all terminate after 15,000 to 50,000 seconds due to a memory error likely caused by some memory limit intrinsic to Paradox’s implementation.

As expected, the results show an exponential increase in runtime with increasing sample size. More interestingly, an exponential increase in runtime can be also be observed the complexity of a definition and not just the number of defined predicates. This is especially noticeable for the definitions of Inc and SC (case 11), which are logically much more
complex than C or PO, as seen from Table 1. C and PO both use a single existentially quantified conjunction, whereas Inc uses a disjunction of two existentially quantified conjunctions, and SC uses an existentially quantified conjunction and a universally quantified disjunction (i.e., resulting from an implication). These differences in logical complexity manifest themselves in additional FOL-CNF clauses with more than 3 literals ($#C_i$; especially for Inc) and additional clauses with 3 variables ($#C$ with $V = 3$ in Table 1; especially for SC). Thus, these measures can likely predict model finding performance.

But based on the six parameters $#C$ with $V = 3$, $#P_v$, $#P_e$, $#C_3$, $#S_v$, and $#S_f$, we expected model finding times for cases 2 to 7 to be relatively close. However, a noticeable difference can be observed when the definition of C is present (cases 3, 4, and 7) vs. when it is absent (cases 2, 5, and 6). These differences are further investigated in Sec. 5.4.

5.3 Exp. 2: Use of Optionally Defined Predicates

We next test whether the presence of additional optional definitions or the use of their optionally defined terms has a greater impact on model finding time. Based on the results from experiment 1, we limited our study to samples with 50 and 60 individuals. For experiment 2, we consider the 4 TBoxes from experiment 1 that add exactly one definitions of either PO, C, Inc, or SC, which are the TBoxes from cases 3 (with C defined), 5 (with PO defined), 8 (with Inc defined), and 11 (with SC defined). For each of the 4 defined predicates we consider two variants of the ABox. In the first one, occurrences of the investigated predicate $\Omega$ are left as-is. In the second one, we alter the ABox from the first one by replacing all occurrences of the $\Omega$ under investigation by its definients, resulting in no use of the defined predicate by the ABox.

The results depicted in Fig. 2 show that using the defined predicates in the ABox does not cause any significant increase in runtime over the second case where the defined predicates are not used. The small increase that is noticeable in some cases (e.g., for PO in sample size 50, or C in sample size 60) is more likely attributable to general variabilty as in other cases a lower mean runtime is measured when the defined predicates are not used. These results confirm that the mere presence of more (defined) terms, rather then their subsequent use in the ABox causes the significant increase in model finding time.

5.4 Exp. 3: Semantic Restrictiveness of Definitions

The results from experiment 2 in Fig. 2 show that the impact of a single definition varies greatly. This can be partly explained by the increasing complexity of these definitions, resulting in more and longer clauses in the propositional representation as measured by the number of extra FOL-CNF clauses with 3 and 2 variables: PO and C result in 1 and 2, Inc in 2 and 9, and SC in 5 and 2 extra clauses with 3 and 2 variables, respectively. This is reflected in an increasing number of propositional clauses as the example for domain size 60 illustrates: 2,268,000 for the cases with PO and C, 2,824,800 for Inc, and 4,316,000 for SC. However, the two cases with PO and C do not differ in any of these metrics, yet model finding with C takes approximately twice as long. The only difference that we are aware of is that PO is semantically more restricted than C, that is, $PO(x, y)$ entails $C(x, y)$ but $C(x, y)$ does not necessarily entail $PO(x, y)$. But since both are defined, their interpretations still fully depends on the primitive predicates, meaning that during model finding for the less restricted definition C, conflicts may go unnoticed longer, resulting in an increase in runtime.

We test this hypothesis by introducing a definition of an artificial new predicate $PO_{\text{new}}$, defined as

$$PO_{\text{new}}(x, y) \iff \exists z[P(x, x) \land \text{Cont}(z, y)]$$

This new definition has the same logical structure as the definitions of C and PO and, thus, generates the same number of variables and clauses in its propositional CNF form (compare to cases 3 and 5 in Table 3). However, $PO_{\text{new}}$ is semantically stronger than C but weaker than PO, that is, for any two individuals $x$ and $y$, $PO(x, y)$ entails $PO_{\text{new}}(x, y)$, which further entails $C(x, y)$. Likewise, $\neg C(x, y)$ entails $\neg PO_{\text{new}}(x, y)$, which in turn entails $\neg PO(x, y)$. We compare the times it takes to find models between PO, $PO_{\text{new}}$, and C. For better comparability, we actually name the predicate PO in all three cases.

The number of facts mentioning C vs. PO differs significantly in the master ABox (compare Table 2). To avoid bias in our comparison, we need equal number of facts for each of the three investigated definitions. We achieve that by using $\text{ABox}_{\{PP,C,PO,Inc,SC\}}$ from case 1 of experiment 1, but adding additional facts about PO to all three cases: (1) all positive facts that use PO – since PO is the most restricted of all three investigated relations – and (2) all negated facts.

Table 4: Median time to find a model in case 1 from experiment 1 for domain sizes 90 to 120.

<table>
<thead>
<tr>
<th>Domain size</th>
<th>90</th>
<th>100</th>
<th>110</th>
<th>120</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time in s</td>
<td>840</td>
<td>1911</td>
<td>4215</td>
<td>8695</td>
</tr>
</tbody>
</table>

Figure 2: Mean time to find a model for (1) when the optional definitions (one of PO, C, Inc, and SC; solid bar) are added to the TBox but any occurrences of the defined terms in the ABox are still replaced by defined assertions, and (2) when the defined predicate is also used by the ABox (hashed bar). Experiments are for samples size 50 and 60.

7 We leave PP out as it did not have much of an impact on model finding time in experiment 1.
The results indicate that model finding performance with \( \text{PO}_{\text{new}} \) is, on average, indeed slightly worse than with PO but better than with C as shown in Figure 3. However, there is lots of variability between samples, with the \( \text{PO}_{\text{new}} \) case in 30% of the samples terminating faster than the PO case. Likewise, the percentage increase in runtime from PO to \( \text{PO}_{\text{new}} \) is much smaller than the increase from \( \text{PO}_{\text{new}} \) to C, the bump at sample size 60 seems more like an artifact from too few samples. This limited study suggests that semantic restrictiveness of definitions somewhat affects model finding performance, though fails to explain the difference between the time for generating models with PO vs. C included as optional definition. Our data suggests that the semantic restrictiveness of definitions has a lesser and a much less predictable impact than the logical complexity of definitions.

6 Conclusions

We presented a first, if small, systematic study of the feasibility of model construction for FOL ontologies with an ABox. We showed that for a mid-sized but fairly complex spatial ontology with significant use of binary predicates, model construction succeeds for up to 40-50 individuals without altering the ontology. But more significantly, removing a single optional definition of a predicate of highest arity – here a binary predicate – and substituting corresponding ABox facts by defined assertion leads to a significant speedup in model finding performance. In our example, the removal of the most complex defined binary predicate (SC) alone led to a speedup of two orders of magnitudes, while removing five of 13 binary predicates resulted in a speedup of three orders of magnitude (over 1,000%) for models with 40 and 50 individuals. The same technique helps scale model construction to models with 120 or more individuals. While not be directly suitable for large-scale reasoning, this at least allows external verification of FOL ontologies using small, yet realistic data samples.

More generally, this demonstrates the gains – both in terms of speed and scalability – that the inexpensive task of removing defined predicates of highest arity can achieve.

Future work must test whether similar, if any at all, speedup can be achieved by the removal of predicates of non-highest arities. Our work outline a viable approach to significantly scale up model finding for ontologies with lots of defined predicates, which can be easily and efficiently implemented as a preprocessing step for existing off-the-shelf model finders, with a corresponding postprocessing step that completes a model with interpretations for the (temporarily removed) defined predicates.

More broadly, this work emphasizes the practical importance of minimizing the number of primitive terms in an ontology, which is often treated as a purely theoretical exercise, but should be a primary concern in the development of FOL ontologies. It facilitates easier and more realistic external verification of ontologies.

Joint Spatial Reasoning Our secondary objective was to test whether existing FOL model finders can be utilized for joint reasoning about general spatial knowledge – as encoded in FOL ontologies of qualitative space – and factual spatial knowledge from geometric maps. This endeavor requires to automatically determine which ontology – from a repository of dozens of spatial ontologies with different terminologies and axiomatic assumptions – a given spatial dataset abides by (i.e., is consistent with), a task that requires proving satisfiability of a TBox (i.e., the "ontology") with an ABox (i.e., the dataset). The so-identified ontology (or ontologies) could provide useful implicit background knowledge about the spatial dataset.

Future Work The presented here is only a first step towards more scalable external verification of FOL ontologies and automated spatial reasoning that leverages FOL ontologies. We intend to test the approach of eliminating optional definitions on a broader range of spatial ontologies and datasets and on other model finders, in particular iProver, which may scale better for larger ontologies especially with predicates of higher arities (i.e., the quaternary betweenness relation from (Hahmann and Grüninger 2011a)).

The variability in the amplitude of the speedup for different definitions suggests taking a closer look at possible factors: (1) SAT-finding heuristics, primarily for variable selection, (2) the complexity of definitions (especially the number and nesting of existential quantifiers as suggested by (Kang, Li, and Krishnaswamy 2012)), and (3) interdependence of terms. For example, variable selection that prioritizes primitives and more general classes/relations over more specialized ones could lead to performance improvements.

A side observation from our work is that the ABox was only a fraction of the number of ground clauses produced from the TBox itself. While scalability for larger domain sizes remains an issue, scalability for larger number of facts (about the same individuals) does not seem much of a concern, suggesting that model finding may scale well for much larger ABoxes as long as the domain size is not increased. That also means that additional, especially negative, facts easily derived from the taxonomic structure of the ontology, could even speed up model finding similarly to how clause learning speeds up SAT solving.
Acknowledgement This material is based in part upon work supported by the National Science Foundation under Grant Number III-1565811.

References