EXPLORING SEMANTIC HIERARCHIES TO IMPROVE RESOLUTION THEOREM PROVING ON ONTOLOGIES

by

Stanley C. Small

A Thesis Submitted in Partial Fulfillment
of the Requirements for a Degree with Honors
(Computer Science)

The Honors College
University of Maine
March 2019

Advisory Committee:

- Dr. Torsten Hahmann, Assistant Professor[1], Advisor
- Dr. Mark Brewer, Professor of Political Science
- Dr. Max Egenhofer, Professor[1]
- Dr. Sepideh Ghanavati, Assistant Professor[1]
- Dr. Roy Turner, Associate Professor[1]

[1]School of Computing and Information Science

ABSTRACT

A resolution-theorem-prover (RTP) evaluates the validity (truthfulness) of conjectures against a set of axioms in a knowledge base. When given a conjecture, an RTP attempts to resolve the negated conjecture with axioms from the knowledge base until the prover finds a contradiction. If the RTP finds a contradiction between the axioms and a negated conjecture, it proves the conjecture.

The order in which the axioms within the knowledge-base are evaluated significantly impacts the runtime of the program, as the search-space increases exponentially with the number of axioms.

Ontologies, knowledge bases with semantic (and predominantly hierarchical) structures, describe objects and their relationships to other objects. For example, a 'Car' class might exist in a sample ontology with 'Vehicle' as a parent class and 'Bus' as a sibling class. Currently, any hierarchical structures within an ontology are not taken into account when evaluating the relevance of each axiom. At present, each predicate is automatically assigned a weight based on a heuristic measure (such as the number of terms or the frequency of predicates relevant to the conjecture) and axioms with higher weights are evaluated first. My research aims to intelligently select relevant axioms within a knowledge-base given a structured relationship between predicates. I will use the semantic hierarchy over predicates to assign weights to each predicate passed to a weighting function. The research aims to design heuristics based upon the semantics of the predicates, rather than solely the syntax of the statements.

I plan to develop weighting functions based upon various parameters relevant to the ontological structure of predicates contained in the ontology, such as the size and depth of a hierarchy based upon the structure of the ontology.

I will implement methods to calculate weights for each predicate and thus each axiom in attempts to select relevant axioms when proving a theorem. Then, I will conduct an experimental study to determine if my methods show any improvements over current reasoning methods.

ACKNOWLEDGEMENTS

Many thanks are given to Dr. Hahmann. This work could not be completed without his continued support and encouragement. Despite his tremendously busy schedule, he always made time to meet and answer questions.

Robert Powell also proved instrumental to the process. He wrote a utility which converts Common Logic Interchange Format (CLIF) into web ontology language (OWL). His work streamlined the testing process and allowed me to find necessary results.

The thesis committee was also instrumental in producing an undergraduate thesis of scale.

List of Figures

1 This sample ontology, with inspiration from an example by Natalya F. Noy, describes entities and relations in the 'automobiles' domain. The ontology serves to provide a "formal explicit description" of classes (outlined in black) along with properties which describe various attributes of the classes (such as how the Subaru Legacy is produced by Subaru). While not displayed in the figure, an ontology also defines property restrictions within the domain (so an instance of the Subaru class cannot produce a car manufacturer). The ontology, along with individual instances of classes (highlighted in red) constitutes a knowledge base [6]. 3

2 Class hierarchies and object property hierarchies for an ontology can be viewed in Protégé [2]. Asserted hierarchies are shown above, and inferred hierarchies are generated by starting the Pellet reasoner. 4

3 The figure above displays a resolution tree for the inference rule described in the previous section. The bold statement shows the negated conjecture. The tree also displays x bound to $SubaruLegacy$. 6

4 Prover9 displays output for the automated proof. 7

5 The figure above illustrates my process of conducting experiments. The first step, converting the ontologies into a formate readable Protégé, lives outside of the scope of my research. 9

6 The GUI for Prover9 on macOS is shown above. One enters axioms into the top text field and conjectures into the bottom text field. Predicate weights are entered in the 'Additional Input' tab in the navigation bar. Users can start the proof by pressing the 'Start' button below the Prover9 logo after specifying a time limit. After the proof has completed, users can view the number of clauses generated by the proof by pressing the 'Info' button below the 'Start' button [5]. 12

List of Tables

1 Results for the multidim_space_voids Ontology 13

2 multidim space voids weights 17

Contents

ABSTRACT **iii**

ACKNOWLEDGEMENTS **iv**

LIST OF FIGURES **v**

LIST OF TABLES **vi**

1 INTRODUCTION **1**

2 BACKGROUND AND RELATED WORK **3**

2.1 Ontologies 3

2.2 Theorem Proving 5

2.3 Semantic Similarity 7

3 APPROACH **9**

3.1 Converting Ontologies 9

3.2 Generating a Complete Heirarchy 10

3.3 Understanding Default Weights 10

3.4 Calculating Weights 11

4 EXPERIMENTS **12**

4.1 Setup 12

4.2 Weighting Functions 13

4.3 Results 13

5 CONCLUSIONS **14**

REFERENCES **15**

A TESTS **17**

AUTHOR'S BIOGRAPHY **19**

1 INTRODUCTION

The rules of logic enable one to prove theorems from axioms stored in a knowledge base. Axioms, asserted facts typically expressed in a formal manner, provide a computer program with tools to confirm or refute conjectures without additional user input. Because computers excel at simple and repetitive tasks, one can witness the applications of automated theorem proving in fields which rely heavily on "knowledge acquisition and information retrieval" [10]. The ability for machines to deduce logically valid conclusions has applications in artificial intelligence and a variety of scientific domains [11]. Automated theorem proving provides a versatile method for reasoning with a set of facts, and has been used to prove and verify proofs of multiple theorems. The four color map theorem, obtained by an automated proof in 1976 by a general-purpose theorem-proving software in 2005 remains a notable example [3]. Moreover, advances have been made in work on the Kepler conjecture and in finding optimal solutions for a Rubik's Cube. The general-purpose nature of automated theorem proving yields applications to a variety of problems. However, automated theorem proving programs often neglect semantic knowledge embedded in an ontology.

Ontologies provide a "common vocabulary" for researchers to speak about a specific domain by describing entities and the relationships between them [6]. A formal description of a specific environment provides researchers and machines with a shared understanding by aiming to capture the semantics of a domain's concepts and relations. Some relationships between an ontology's terms may be explicitly defined, but many are implicit. Below, one can find two sample axioms he or she might witness in a knowledge base.

$$isSedan(SubaruLegacy)$$
$$\forall x\ isSedan(x) \rightarrow isAutomobile(x)$$

The former axiom asserts a Subaru Legacy is a sedan and the latter asserts all sedans are automobiles. While one can easily deduce the fact a Subaru Legacy is an automobile, no single axiom explicitly describes such a statement. Fortunately, formal logic defines rules of inference

which allow one to transform established facts into new conclusions solely based on the syntax of these statements. Both humans and computers can clearly distinguish well formed statements ($x + y = 4$) from those which are not ($x4y+ =$). Beyond the syntax of statements, ontologies and logics also define the semantics or meaning of sentences (i.e. declaring $x + y = 4$ is true when $x = 1$ and $y = 3$ but false when $x = 0$ and $y = 1$).

Like many taxonomies, or schemes of classification, one can often form maps of relationships within an ontology which resemble a hierarchy. Knowledge encoded in semantic hierarchies can help an automated theorem prover determine which axioms might be most helpful when attempting to prove a specificconjecture. This work attempts to improve automated theorem proving with ontologies by identifying relevant facts, and ignoring those less likely to yield a proof.

2 BACKGROUND AND RELATED WORK

2.1 Ontologies

The word ontology ("study of being") combines Greek *onto-* ("being") and *-logia* ("logical discourse"). The act of studying knowledge and existence in philosophy has given birth to the study of formal logic and automated reasoning in computer science. Researchers often use ontologies to share information among people or computer programs, to enable domain knowledge reuse, to make definitions of a particular domain explicit, or to analyze domain knowledge [6]. While ontologies can often witness reuse, ontology design rarely encounters a single best solution satisfying all goals of the ontology.

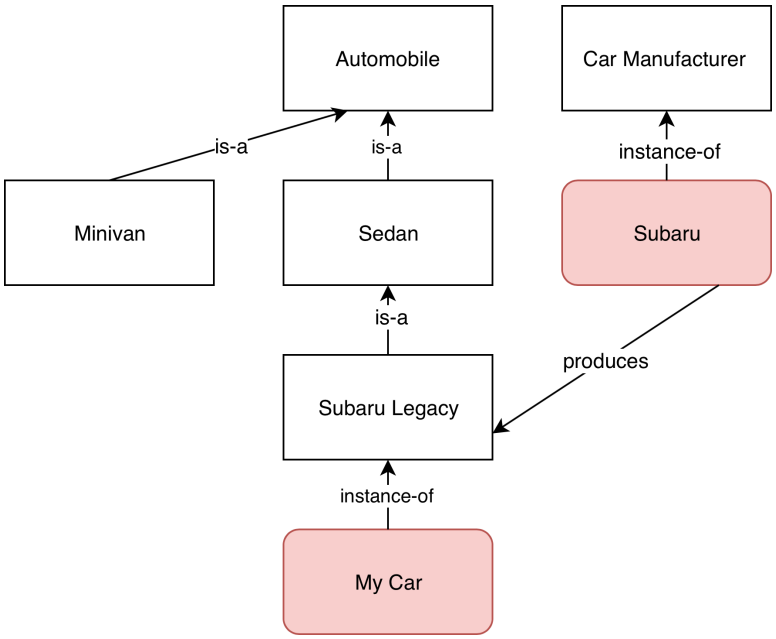

Figure 1: This sample ontology, with inspiration from an example by Natalya F. Noy, describes entities and relations in the 'automobiles' domain. The ontology serves to provide a "formal explicit description" of classes (outlined in black) along with properties which describe various attributes of the classes (such as how the Subaru Legacy is produced by Subaru). While not displayed in the figure, an ontology also defines property restrictions within the domain (so an instance of the Subaru class cannot produce a car manufacturer). The ontology, along with individual instances of classes (highlighted in red) constitutes a knowledge base [6].

In reality, few differences between an ontology and a knowledge base exists. Knowledge engineers

must traverse a "fine line where the ontology ends and the knowledge base begins" [6]. At the least, an ontology defines categories (or classes) and relationships among objects. One can think of an ontology as a "vocabulary" used to describe a domain [9, p. 308]. Typically, the classes can be arranged in a hierarchy and the object properties into a separate hierarchy, which are here referred to as semantic hierarchies. When designing an ontology, one must decide the scope and organization of the knowledge, along with the language used to describe a specific domain. Refer to Figure 1 for an example.

RELATIONS (subclass and subproperty, domain and range restrictions)

2.1.1 Class and Object Property Inheritance

EXPLAIN the terms and they structure knowledge and how inheritance works

TRANSITION

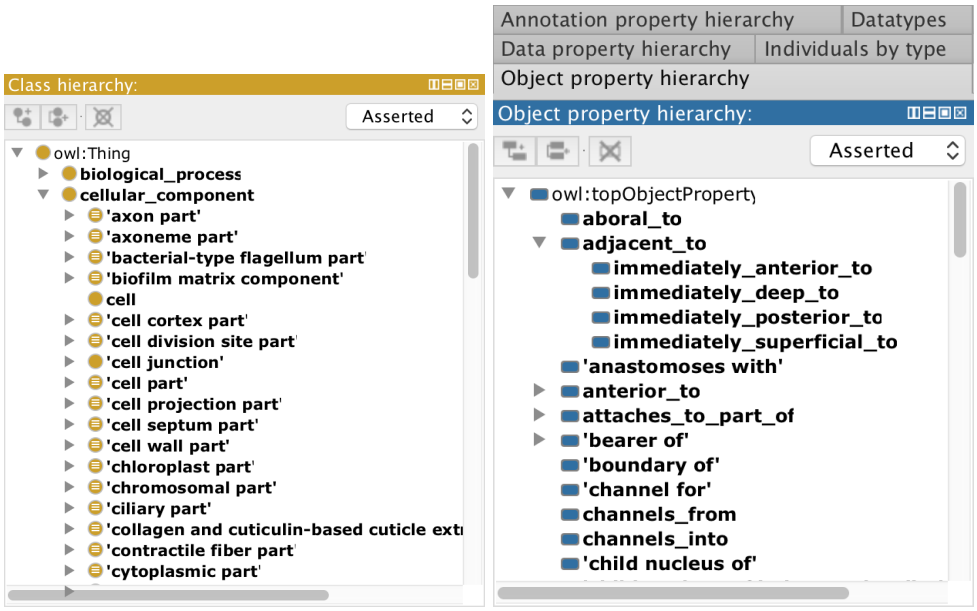

Figure 2: Class hierarchies and object property hierarchies for an ontology can be viewed in Protégé [2]. Asserted hierarchies are shown above, and inferred hierarchies are generated by starting the Pellet reasoner.

EXAMPLE how the semantic hierarchy may help in a proof (when there are lots of irrelevant axioms).

2.2 Theorem Proving

Automated theorem proving depends on having an established logic for expressing facts, a method of generating new facts without requiring additional knowledge, and a strategy for searching through all possible new facts one could generate to reach a specific goal (such as proving a conjecture).

2.2.1 First-order Predicate Logic

Automatic theorem proving requires a logic defining the syntax of valid statements to run without additional user input. Formal logics like first-order logic, also known as predicate logic and first-order predicate calculus, define a structure for statements which can be used to form logical and mathematical proofs. Consider the following set of asserted facts expressed in first-order predicate logic, commonly referred to as axioms.

$$isSedan(SubaruLegacy)$$
$$\forall x\ isSedan(x) \rightarrow hasFourSeats(x)$$

The first statement asserts the Subaru Legacy is a sedan. The second statement asserts all sedans have four seats. By expressing facts in a formal notation, one makes proofs using such statements mechanical and easily parsed by a computer. Expressing statements in formal logic poses multiple challenges. First, each object and relationship must be explicitly defined.

EXPLAIN FIRST ORDER PREDICATE LOGIC

WHAT is a predicate

2.2.2 Inference Rules

One can use axioms to derive facts which logically follow using inference rules. The two previous statements do not directly state the Subaru Legacy has four seats. However, one can derive the

statement $hasFourSeats(SubaruLegacy)$ by using the inference rule *modus ponens*, defined below.

$$\begin{gathered} A \\ A \to B \\ \therefore B \end{gathered} \tag{1}$$

One can think of A and B as variables representing statements, and any statement can replace them. An inference rule defines a valid rule for statements.

2.2.3 Resolution

Automated theorem proving requires a set of axioms and a set of rules to generate new facts, but also a strategy to search through the possible applications of the inference rules. Knowledge bases can grow quite large, and generating all possible facts based on a given set of axioms often remains impractical or unfeasible. Resolution exists as historically significant and widely used method for automated theorem proving [1, p. 51].

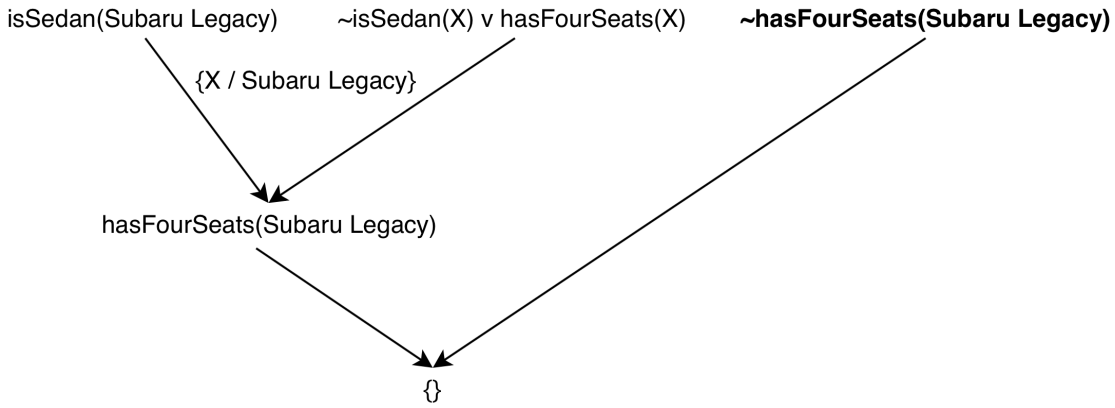

Figure 3: The figure above displays a resolution tree for the inference rule described in the previous section. The bold statement shows the negated conjecture. The tree also displays x bound to $SubaruLegacy$.

In order to use resolution as a proof technique, axioms must first be expressed in Conjunctive Normal Form (CNF), also known as clausal form. One may follow a 7-step procedure of converting the set of facts into a conjunction of disjunctions. The process eliminates biconditionals, implications, and quantifiers so the second axiom $\forall x\ isSedan(x) \to hasFourSeats(x)$ becomes

$\neg isSedan(x) \vee hasFourSeats(x)$. One can then resolve the statements by instantiating the variable x with $SubaruLegacy$ in a process called unification, binding the variable.

$$\frac{isSedan(SubaruLegacy), \neg isSedan(x) \vee hasFourSeats(x)}{hasFourSeats(SubaruLegacy)}$$

Finally, one can resolve the axiom with the negated conjecture, proving the statement true.

```
1 (all x (isSedan(x) -> hasFourSeats(x))) # label(non_clause).  [assumption].
2 hasFourSeats(SubaruLegacy) # label(non_clause) # label(goal).  [goal].
3 -isSedan(x) | hasFourSeats(x).  [clausify(1)].
4 isSedan(SubaruLegacy).  [assumption].
5 hasFourSeats(SubaruLegacy).  [resolve(3,a,4,a)].
6 -hasFourSeats(SubaruLegacy).  [deny(2)].
7 $F.  [resolve(5,a,6,a)].
```

Figure 4: Prover9 displays output for the automated proof.

As a knowledge base grows, the number of clauses an automated theorem prover can generate greatly increases. Researches have begun to form heuristics to evaluate the relevance of axioms when completing a proof.

2.3 Semantic Similarity

Evaluating the similarity of two entities, or predicates when using a theorem prover, can serve as a heuristic for reducing the number of clauses generated during a proof. Multiple metrics have been developed for evaluating the semantic similarity of terms with different approaches, including: edge-counting measures, feature-based measures, and measures based on Information Content [10] [**rodriguez1999assessing**] [8]. Edge-counting metrics for semantic similarity when applied to semantic hierarchies remain the focus of this work.

Calculating the distance between two entities in an ontology is a a straightforward and intuitive method of calculating the semantic similarity. One can formally define the metric as follows. In an undirected graph G defined as a pair (V, E), where V is a set of vertices, and E is a set of edges between the vertices $E \subseteq (u, v)|u, v \in V$, one can define a path $path(a, b) = l_{1,\ldots,}l_k$ as a set of links connecting a and b in a taxonomy and $|path(a, b)| = k$ as the length of the path [10]. One can

calculate the semantic distance between a and b using equation 2 [7].

$$sim(a,b) = min_{\forall i}|path_i(a,b)| \tag{2}$$

By incorporating depth of the taxonomy into the function, improvement has been seen [12].

$$sim(a,b) = \frac{2 \times N_3}{N_1 + N_2 + 2 \times N_3} \tag{3}$$

3 APPROACH

This work aims to evaluate the effectiveness of using a semantic hierarchy generated from an ontology to calculate weights for predicates based on a specific conjecture. In efforts to quantitatively evaluate the effectiveness of the proposed methods, a series of experiments were conducted on multiple ontologies from the COmmon Logic Ontology REpository (COLORE). The objective of COLORE is to serve as a "testbed for ontology evaluation and integration techniques" [4]. Code from the project can be found at `https://code.google.com/p/colore`. Protégé, an "open-source ontology editor and framework for building intelligent systems" was used to generate the semantic hierarchies. The ontologies had to first be converted into a format Protégé can read. Then, Protégé, along with a semantic reasoner, were used to generate both a class hierarchy and an object property hierarchy for each ontology. The semantic hierarchies were then used to calculate the assigned weights for each predicate when executing proofs. Finally, tests were run using Prover9 to compare the default weights to the calculated weights. The effectiveness of the process was evaluated by comparing the number of clauses generated by Prover9 for each proof.

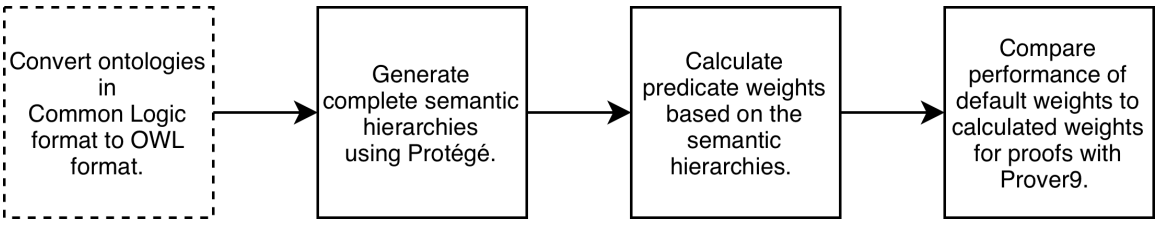

Figure 5: The figure above illustrates my process of conducting experiments. The first step, converting the ontologies into a formate readable Protégé, lives outside of the scope of my research.

3.1 Converting Ontologies

Dr. Michael Grüinger specified ontologies contained in COLORE using Common Logic. One must convert the ontologies from the Common Logic Interchange Format (CLIF) to the Web Ontology Language (OWL) in order to view the ontology in Protégé and generate the semantic hierarchies. Robert Powell has written a utility which executes the conversion and has generated the files neces-

sary to conduct this research.

COLORE contains the knowledge base for each ontology. Additionally, one finds the conjectures used to test each ontology at `https://github.com/gruninger/colore/tree/master/ontologies`. Not all ontologies in COLORE have conjectures, which limits the scope of my experiments to the sufficiently large ontologies with multiple defined conjectures to test.

3.2 Generating a Complete Heirarchy

After opening an ontology defined using the OWL format in Protégé, one can generate both a class hierarchy and an object property hierarchy for including both asserted relationships and inferred relationships. Protégé generates a view of inferred semantic hierarchies (shown in Figure 2) using a semantic reasoner called Pellet. The reasoner parses the axioms for inferred logical consequences (i.e. relationships between classes) not explicitly defined. The reasoner generates more complete and accurate semantic hierarchies.

Given an ontology and a conjecture, one can generate weights for classes and properties to reduce the number of clauses generated in a proof. Functions are evaluated by their influence on the number of clauses generated with a proof.

After a hierarchy has been generated, weights can be assigned to each class and subproperty. The same weighting function is applied to both the classes and sub-properties.

The weighting functions are currently applied by hand to the ontologies, with the beginnings of an automated program underway.

3.3 Understanding Default Weights

Prover9 assigns weights to predicates automatically unless the user explicitly defines them. An understanding of the process helps one to develop new weights for the predicates. Rules for weighting axioms in terms of relevance when attempting to prove a specific conjecture are as follows [5]:

- The default weight of a constant or variable is 1.
- The default weight of a term or atomic formula is one more than the sum of the weights of its arguments.

- The default weight of a literal is the weight of its atomic formula.
- The default weight of a clause is the sum of the weights of its literals.

One can assign weights to predicates explicitly,

3.4 Calculating Weights

Two explicit weighting functions inspired by related works in calculating semantic similarity, were formed and tested. The weights for each conjecture were then calculated by hand and entered into a spreadsheet. A python script was used to generate the input files used by Prover9 from the spreadsheet for each conjecture. The calculated weights were then entered into Prover9 as additional input along with the axioms and the conjecture.

4 EXPERIMENTS

4.1 Setup

Experiments were conducted using Prover9, an "automated theorem prover for first-order and equational logic" written by William McCune [5]. Many tests were conducted using a version of the program which supports a Graphical User Interface (GUI), but a command line version, useful for running automated tests, exists. Git was used for version control and a repository containing source code for tests can be found at `https://github.com/stanleysmall/thesis`.

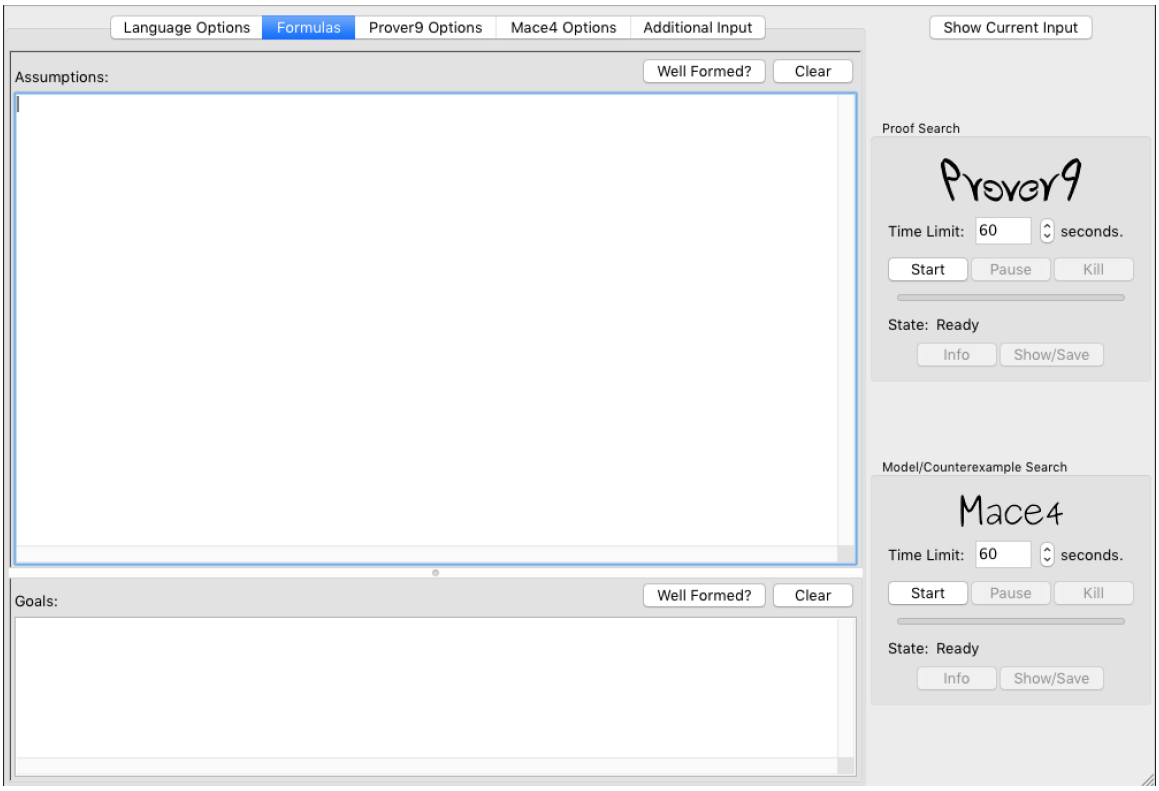

Figure 6: The GUI for Prover9 on macOS is shown above. One enters axioms into the top text field and conjectures into the bottom text field. Predicate weights are entered in the 'Additional Input' tab in the navigation bar. Users can start the proof by pressing the 'Start' button below the Prover9 logo after specifying a time limit. After the proof has completed, users can view the number of clauses generated by the proof by pressing the 'Info' button below the 'Start' button [5].

4.2 Weighting Functions

Two weighting functions were defined for testing.

4.3 Results

4.3.1 multidim_space_voids

Goal	Default	Function 1	Change 1	Function 2	Change 2
1	100	80	-20	60	-40
2					
3					
4					
5					
6					
7					
8					
9					

Table 1: Results for the multidim_space_voids Ontology

5 CONCLUSIONS

In many cases, the algorithm increases the number of clauses generated for each test, but does not do so to the point where the tests are unusable. For some very specific ontologies, the number of clauses generated decreases. This can be attributed to, in part, by the small number of ontologies available for testing, along with the specific pattern and hierarchy of each ontology.

Many opportunities for further research include fully automating the search procedure, working with a larger number of ontologies to ensure the weighting functions actually do as they say, developing a new approach towards automatically weighting the predicates.

References

[1] Wolfgang Ertel. *Introduction to artificial intelligence*. Springer, 2018.

[2] John H Gennari et al. "The evolution of Protégé: an environment for knowledge-based systems development". In: *International Journal of Human-computer studies* 58.1 (2003), pp. 89–123.

[3] Georges Gonthier. "Formal proof–the four-color theorem". In: *Notices of the AMS* 55.11 (2008), pp. 1382–1393.

[4] Michael Grüninger and Megan Katsumi. "Specifying ontology design patterns with an ontology repository". In: *Proceedings of the 3rd International Conference on Ontology Patterns-Volume 929*. Citeseer. 2012, pp. 1–12.

[5] William McCune. *Prover9 and mace4*. 2005.

[6] Natalya F Noy, Deborah L McGuinness, et al. *Ontology development 101: A guide to creating your first ontology*. 2001.

[7] Roy Rada et al. "Development and application of a metric on semantic nets". In: *IEEE transactions on systems, man, and cybernetics* 19.1 (1989), pp. 17–30.

[8] Alex Roederer, Yury Puzis, and Geoff Sutcliffe. "Divvy: An ATP meta-system based on axiom relevance ordering". In: *International Conference on Automated Deduction*. Springer. 2009, pp. 157–162.

[9] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited, 2016.

[10] David Sánchez et al. "Ontology-based semantic similarity: A new feature-based approach". In: *Expert systems with applications* 39.9 (2012), pp. 7718–7728.

[11] Josef Urban. "An overview of methods for large-theory automated theorem proving". In: (2011).

[12] Zhibiao Wu and Martha Palmer. "Verbs semantics and lexical selection". In: *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics. 1994, pp. 133–138.

Table 2: multidim space voids weights

Entity	Type	Superclass(es)	1	2	3	4	5	6	7	8	9
CAVITY	Class	owl:Thing	10								
Closed	Class	owl:Thing	10								
ComplexV	Class	V	2								
Con	Class	S									
DPF	Class	F									
F	Class	"PEDΨ RPForDPF"									
Gap	Class	owl:Thing	2								
HOL	Class	owl:Thing	2								
Hole	Class	owl:Thing	10								
ICon	Class	Con									
M	Class	"PEDΨ mat"									
Max	Class	S									
MaxDim	Class	S									
Min	Class	S									
MinDim	Class	S									
NAPO	Class	POB									
PED	Class	POBorMorF	10								
POB	Class	PED mat									
POBorMorF	Class	owl:Thing									
POBorMorRPF	Class	owl:Thing									
POBorRPF	Class	owl:Thing									
RPF	Class	F mat									
RPForDPF	Class	owl:Thing									
S	Class	owl:Thing	10								
SimpleV	Class	V	2								
SimpleVorComplexV	Class	owl:Thing									
TUN	Class	owl:Thing	10								
V	Class	SimpleVorComplexV	1								
ZEX	Class	S									
mat	Class	POBorMorRPF									
BCont	ObjectProperty										
C	ObjectProperty										
Cont	ObjectProperty										
Covers	ObjectProperty										
DK1	ObjectProperty										
EqDim	ObjectProperty										
ICont	ObjectProperty										
Inc	ObjectProperty										
P	ObjectProperty										
PO	ObjectProperty										
PP	ObjectProperty										
SC	ObjectProperty										
TCont	ObjectProperty										
VS	ObjectProperty										
ch	ObjectProperty										
gt	ObjectProperty										
hosts	ObjectProperty										

Entity	Type	Superclass(es)	1	2	3	4	5	6	7	8	9
hostscavity	ObjectProperty										
hostscavityi	ObjectProperty										
hostscavityt	ObjectProperty										
hostsg	ObjectProperty										
hostsh	ObjectProperty										
hostshollow	ObjectProperty										
hoststunnel	ObjectProperty										
hostsv	ObjectProperty										
hostsve	ObjectProperty										
hostsvi	ObjectProperty										
lt	ObjectProperty										
r	ObjectProperty										
”=”	ObjectProperty										
”gt=”	ObjectProperty										
”lt=”	ObjectProperty										

AUTHOR'S BIOGRAPHY

Stanley C. Small grew up in Hampden, Maine with his mother Diane and his father Scott. He attended the University of Maine and received a Bachelor of Science degree in Computer Science in May of 2019.