Automatic White-Box Testing of First-Order Logic Ontologies

JAVIER ÁLVEZ, MONTSERRAT HERMO, PAQUI LUCIO, GERMAN RIGAU
Facultad de Informática, University of the Basque Country UPV/EHU,
Paseo Manuel de Lardizabal, 1, 20018-San Sebastián, Spain.

Formal ontologies are axiomatizations in a logic-based formalism. The development of formal ontologies is generating considerable research on the use of automated reasoning techniques and tools that help in ontology engineering. One of the main aims is to refine and to improve axiomatizations for enabling automated reasoning tools to efficiently infer reliable information. Defects in the axiomatization can not only cause wrong inferences, but can also hinder the inference of expected information, either by increasing the computational cost of, or even preventing, the inference.
In this paper, we introduce a novel, fully automatic white-box testing framework for first-order logic (FOL) ontologies. Our methodology is based on the detection of inference-based redundancies in the given axiomatization. The application of the proposed testing method is fully automatic since a) the automated generation of tests is guided only by the syntax of axioms and b) the evaluation of tests is performed by automated theorem provers. Our proposal enables the detection of defects and serves to certify the grade of suitability –for reasoning purposes– of every axiom. We formally define the set of tests that are (automatically) generated from any axiom and prove that every test is logically related to redundancies in the axiom from which the test has been generated. We have implemented our method and used this implementation to automatically detect several non-trivial defects that were hidden in various first-order logic ontologies. Throughout the paper we provide illustrative examples of these defects, explain how they were found, and how each proof –given by an automated theorem-prover–provides useful hints on the nature of each defect. Additionally, by correcting all the detected defects, we have obtained an improved version of one of the tested ontologies: Adimen-SUMO.

Keywords: Automated Theorem Proving, Fisrt-Order Logic, Ontology, Knowledge representation, White-box testing.

1. Introduction

Formal ontology development [46, 25, 23, 56] is a discipline whose goal is to represent explicit formal specifications (axiomatizations) of terms in a domain and relations between them. Many research areas –such as Semantic Web, Knowledge Representation, Commonsense Representation and Reasoning [38, 40, 15]– have converged on the adoption of formal ontologies as explicit conceptualizations that are able to support automated reasoning. In this paper we focus on First-Order Logic (FOL) ontologies. Description Logic (DL) [10] is a family of formal knowledge representation languages that is very commonly used in the ontology area, but DL is less expressive than FOL. Ontologies such as e.g. CYC [37], DOLCE [20] and SUMO [44] which are very useful in some areas –e.g. commonsense reasoning and natural language processing– use more expressive languages (FOL or higher). Moreover, DL is a sublanguage of FOL and consequently the techniques presented in this paper can also be applied to DL ontologies.[1]

As with other software artifacts, ontologies have to fulfill some previously specified requirements. Clear (and explicit) ontological distinctions and principles such as those provided by OntoClean [25] reduce the risk of classification mistakes in the ontology development process, and can simplify its maintenance. Both the creation of ontologies and the verification of its requirements are usually semi-automatic tasks that require a significant amount of human effort [2]. Once the ontology is stable and consistent (or at least no inconsistencies can be found by automatic means), the methods for ontology testing can be classified into two main categories: *black-box* and *white-box*, as in the field of software testing [42]. Black-box methods are based on the use of tests defined according to the requirements of the given ontology, while white-box methods are characterized by the fact that tests are created upon the particular specification/codification of the knowledge. In this paper, every ontology to be tested is considered to be the specification or codification, and a set of tests is automatically constructed from the axioms of the considered ontology. The construction of each test depends only on the syntactical form of an axiom. In this sense, our strategy is white-box. As far as we know, the only testing method (for FOL ontologies) that can be classified as white-box is proposed in [53], where the authors propose to create potentially unsatisfiable subsets of axioms by applying SInE strategies [29] on the basis of a selected *seed* symbol. In this way, large first-order knowledge bases can be proved to be inconsistent if some inconsistent subset is found. Among black-box testing methods, most frequent ones are based on consistency checking. State-of-the-art reasoners such as FaCT++ [62], Pellet [55] or HermiT [22] enable consistency proving in the case of DL ontologies. On the contrary, proving the consistency of ontologies expressed in FOL such as SUMO or DOLCE is much harder. Additionally, ontology testing methods include those based on the use of *competency questions* (CQs) [24] for validating functional requirements. That is, the *competency* of an ontology is described by means of a set of goals or problems that are expected to be answers according to its requirements. The process of obtaining CQs is not automatic but creative [17]. Depending on the size and complexity of the ontology, creating a suitable set of CQs is by itself a very challenging and costly task. In [3, 4], we propose a method for the semi-automatic creation of CQs for SUMO-based FOL ontologies on the basis of WordNet

[1] Indeed, in this paper we report on the application of our techniques to the DL ontology DOLCE-CASL.

[16] and its mapping into SUMO [45]. Alternatively, in [13] the authors propose a tool that creates and processes CQs written in natural language for OWL ontologies. Finally, some other testing methods are based on cross-checking ontologies against another knowledge bases. For example, the authors of [48] propose to find errors in DBpedia [9, 14] by its alignment to DOLCE, and contextual knowledge extracted from DBpedia is used for detecting hidden errors in DL ontologies as described in [61].

Regarding ontology debugging methods, the classification is slightly different [47]: *black-box* methods use reasoners as the oracle for a certain set of questions e.g., subsumption, satisfiability, etc.; *glass-box* methods are based on information extracted from the internals of reasoners, which are sometimes specifically adapted for the debugging task. There exists a large variety of techniques which are used in both classes of methods for DL ontologies. Among others, justification based techniques in black-box methods [19, 31, 30, 54], axiom pinpointing in black-box [12] and glass-box methods [11], and checking unsatisfiable dependant paths in glass-box methods [64]. Further, black- and glass-box methods are combined in some proposals, using both axiom pinpointing [52] and justification based techniques [32].

In [24], the authors propose a methodology for the design and evaluation of ontologies on the basis of a set of CQs. The only requirement for applying the proposed methodology is the existence of a decision algorithm for the underlying logic. An adaptation of this methodology to FOL ontologies is introduced in [3], which enables ontologies to be automatically evaluated by means of the use of FOL Automated Theorem Provers (ATPs). In this adaptation, the set of CQs is partitioned into two sets: *truth-tests* and *falsity-tests*, depending on whether one expects the conjecture to be entailed by the ontology (*truth-tests*) or not (*falsity-tests*). An example of truth-test is *“Siblings have the same mother”*, since it is expected to be entailed by an ontology (Of course, the ontology should axiomatize the involved concepts). This truth-test belongs to the *Commonsense Reasoning* CSR domain of the *Thousands of Problems for Theorem Provers* (TPTP) problem library[2] [57] and the latter one is a CQ in the benchmark proposed in [3]. On the contrary, the conjecture *“Some herbivores eat animals”* is a falsity-test since it is not expected to be entailed by the ontology, in spite that the ontology should axiomatize the involved concepts.

In this paper, we introduce a completely automatic methodology for the evaluation of FOL ontologies by means of a set of automatically generated falsity-tests and truth-tests. We define the logical foundations of these sets of tests and prove the correctness of the methodology. We also describe the application of our methodology to DOLCE [21], FPK (formal proof of the Kepler conjecture) [27] and Adimen-SUMO [2]. In particular, we review in detail the kind of defects that have been detected in

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

those ontologies. An improved version of Adimen-SUMO v2.6 has been obtained by correcting all the defects detected following the introduced methodology. An example of incorrect axiom that we have detect in Adimen-SUMO v2.4 following our methodology is part of the axiomatization of the relation *sibling*. The wrong axiom asserts that *"Any two members of the same broad are siblings"*, instead of "Any two *different* members of the same broad are siblings" (see Subsection 7.4 for details). The latter is the correct version of the axiom in Adimen-SUMO v2.6.

The paper is organized as follows. First, we briefly describe the ontologies DOLCE, FPK and Adimen-SUMO in Section 2. In Section 3, we introduce the proposed methodology for evaluating ontologies utilizing ATPs. Next, in Section 4, we formally define the set of tests that are proposed for a given axiom. Then, in Section 5, we provide a detailed example illustrating the calculation of tests. In Section 6, we prove the correctness of the proposed set of tests. In Section 7, we report on the main kinds of defects that we have found in the evaluated ontologies, explaining specific examples of four different types. We provide a summary of our experimental results in Section 8. Finally, we give some conclusions and discuss future work.

2. FOL Ontologies

FOL formulas are constructed on an alphabet (or signature) of function and predicate symbols, using the logical connectives of negation ($\neg$), conjunction ($\wedge$), disjunction ($\vee$), implication ($\rightarrow$) and double-implication ($\leftrightarrow$) , as well as the universal and existential quantifiers (resp. $\forall$ and $\exists$). The notation $\overline{Qx}$ stands for a sequence of quantifiers (on variables) $Q_1x_1 \ldots Q_nx_n$ such that $n \geqslant 0$ and $Q_i \in \{\exists, \forall\}$ for each $1 \leqslant i \leqslant n$. In the FOL formulas in this paper, functions symbols (in particular, constants) start with a capital letter, whereas predicates start with a lower-case and variables are lower-case letters (possibly with subindices). We use *sentence* to refer to any FOL formula for which all its variable occurrences are in the scope of (or bound by) a quantifier. We assume that the reader has some familiarity with the syntax and basic notions of FOL.

FOL ontologies consist of a set of FOL sentences called *axioms*. Typically, ontology axioms are classified into *rules* and *non-rules*. Rules are universally closed implications. An example of rule axiom in DOLCE is (1). It is also common to call the left-hand part of the implication *antecedent*, and the right-hand part *consequent*. In this section, we introduce the main features of the FOL ontologies that have been used for the evaluation of our white-box testing strategy —DOLCE, FPK and Adimen-SUMO— and provide some figures about their content. We also provide some examples of axioms and tests in the form of FOL sentences. Each FOL ontology uses its own alphabet of functions and predicates, named with strings (e.g. *world*, *mother*, *agent*, ...) that usually try to express its meaning, but their formal meaning is given by their FOL axiomatization in the ontology.

DOLCE (*Descriptive Ontology for Linguistic and Cognitive Engineering*) is the first-module of a *Library of Foundational Ontologies* being developed within the WonderWeb project [21]. Its main purpose is supporting effective cooperation between multiple artificial agents and establishing consensus in a mixed society where artificial agents cooperate with human beings. The partial mappings from WordNet [16] to DOLCE [20] enable the connection of DOLCE to other semantic resources such as the Multilingual Central Repository (MCR) [8] and its application to advanced Natural Language Processing, Knowledge Engineering and Semantic Web tasks [63]. The domain of discourse of DOLCE is restricted to the notion of *particulars* —entities which have no instances. Similarly, particulars are characterized and organized around a taxonomy of 37 *universals*—entities that can have instances— and universals are organized around the notion of *world*. However, no particular and no world is explicitly defined. DOLCE is originally expressed in KIF according to the standard proposed in [33][3] and simplified translations into various logical languages have been proposed (DOLCE-Lite-Plus[4]). The KIF version of DOLCE uses row variables —which produces variable-arity relations— and quantified predicate symbols. Hence, we have applied the translation described in [2] for its transformation into a pure FOL formula (from now on, KIF-DOLCE). As result, we have obtained 257 rule-axioms such as the following:

$$\forall w\, \forall f\ (\ (universal(f) \wedge world(w)) \rightarrow nep(w,f)\ ) \tag{1}$$

where $nep(w,f)$ stands for the non-emptiness of the universal f in the world w. Vampire v4.1 [51, 34] proves that KIF-DOLCE is consistent. In addition, a simplified translation of DOLCE into CASL [7] (from now on, CASL-DOLCE) is available in Hets [41] and its consistency is proved in [36]. This translation consists of 416 non-atomic formulas such as

$$\exists y\ (\ pED(y)\ ) \tag{2}$$

where *pED* is used to state that the universal *Endurant* has some particular.[5] Both FOL versions of DOLCE —KIF-DOLCE and CASL-DOLCE— were expected to be non-defective due to their reduced size and their mature state of development.

FPK (*formal proof of the Kepler conjecture*) is an ontology that has been derived from the Flyspeck project [27] for its use in the *CADE ATP System Competition CASC-J8* [58]. The purpose of the Flyspeck project is to give a formal proof of the Kepler conjecture, which asserts that no packing of congruent balls in three-dimensional Euclidean space has a density greater than that of the face-centered cubic packing [26].

[3] `http://logic.stanford.edu/kif/dpans.html`

[4] `http://www.loa.istc.cnr.it/old/DOLCE.html`

[5] 37 formulas like (2) are used in CASL-DOLCE for stating the property in formula (1).

Its participants claim that it "is the most complex formal proof ever undertaken" and estimate that it may take about twenty working-years to complete the formalization. To this end, every logical inference is checked against the foundational axioms of mathematics with the help of a computer and, no matter how trivial they are, no step is skipped. We have used the version of FPK that was provided for the CASC competition, since it already consists in a pure FOL axiomatization with 78,500 axioms, such as the following:

$$\forall a\, \forall x\, \forall y\ (\ s(a,x) = s(a,y)\ \rightarrow\ s(a,y) = s(a,x)\ )$$

Though FPK is much larger than DOLCE, FPK has also reached a very mature state of development. Consequently, we did not expect to discover many defects in FPK either.

Finally, Adimen-SUMO has been derived from SUMO (*Suggested Upper Merged Ontology*)[6] [44], which was promoted by a group of engineers from the IEEE Standard Upper Ontology Working Group as a formal ontology standard during the nineties of the past century. Their goal was to develop a standard upper ontology to promote data interoperability, information search and retrieval, automated inference and natural language processing. SUMO is expressed in SUO-KIF (*Standard Upper Ontology Knowledge Interchange Format* [49]), which is a dialect of KIF, and its syntax goes beyond FOL. Consequently, SUMO cannot be directly used by FOL ATPs without a suitable transformation. Furthermore, in order to support higher-order aspects, a translation of SUMO is also required for its use by means of pure higher-order theorem provers [50]. In [2], the authors use ATPs for reengineering around 88% of the top and the middle levels of SUMO into Adimen-SUMO,[7] which can be expressed as a FOL formula. This translation is based on a small set of meta-predicates —and its axiomatization— that are required to define the knowledge of SUMO according to its organization around four kinds of concepts: *objects*, *classes*, *relations* and *properties*. Some of these meta-predicates are *\$instance*, *\$subclass*, *\$disjoint* and *\$partition*.[8] In Adimen-SUMO, like in other ontologies expressed in KIF (e.g. SUMO or KIF-DOLCE), *instance* is used to assert that an object is in a class. For example, the atom $instance(h, Herbivore)$, instead of $Herbivore(h)$, is used to express that object h is in the class $Herbivore$. In addition, [2] provides a suitable translation of domain (or type) information of relations which, in SUMO, is (separately) provided by means of *domain* axioms. For example, in Adimen-SUMO, there are four non-rule axioms asserting that the first argument of predicate *\$instance* is an object, whereas the second one is a class, and that both arguments of *\$subclass* are classes. Adimen-SUMO has also three rules for axiomatizing

[6] `http://www.ontologyportal.org`

[7] The first version of Adimen-SUMO is v2.2.

[8] In Adimen-SUMO, meta-predicates names are marked with a $. In this paper, we omit these marks since ATPs deal with them like any other predicate symbol in the alphabet.

$subclass as a partial order, i.e. each rule respectively says that *$subclass* is reflexive, antisymmetric and transitive. Additionally, the following rule axiomatizes *$instance* in terms of *$subclass*:

$$\forall x \, \forall y \, \forall z \; (\; (\; \$instance(x,y) \; \wedge \; \$subclass(y,z) \;) \; \rightarrow \; \$instance(x,z) \;).$$

Many other meta-predicates are defined in terms of *$subclass* and *$instance*. For example, the predicate *$disjoint* is defined by the following rule in Adimen-SUMO:

$$\forall x \, \forall y \; (\; \$disjoint(x,y) \; \leftrightarrow \; \forall z \; (\; \neg\$instance(z,x) \; \vee \; \neg\$instance(z,y) \;) \;).$$

The interested reader is referred to [2] for a detailed description of the axiomatization and the translation.

Adimen-SUMO —like DOLCE— also uses row variables and quantified predicate symbols. In [3], we introduce an evolved version of Adimen-SUMO (namely, v2.4) and demonstrate its inference capabilities in practice. More specificaly, we exploit the whole mapping of WordNet to SUMO [45] in order to obtain a set of CQs by following a black-box testing strategy. As reported in [3], we have experimentally tested Adimen-SUMO v2.4 using the resulting set of CQs and no defect has been detected. Additionally, we have used Adimen-SUMO v2.4 and the same set of CQs for an experimental comparison of several FOL ATPs in [5]. The state of development of Adimen-SUMO v2.4 was not mature and we were able to detect various defects by following the white-box testing methodology introduced in this paper. As result of correcting all the detected defects, we obtained Adimen-SUMO v2.6, which has been already used in the experimentation reported in [1, 6].

Table 1: Some figures about the evaluated ontologies

Ontology	Non-rules	Rules	Total
KIF-DOLCE	0	257	257
CASL-DOLCE	0	416	416
FPK	275	78,225	78,500
Adimen-SUMO v2.4	4,635	2,785	7,420
Adimen-SUMO v2.6	4,638	2,799	7,432

In Table 1, we summarize some figures about DOLCE, FPK and Adimen-SUMO (v2.4 and v2.6): the number of (non-rule and rule) axioms that result from their transformation into a pure FOL formula (no transformation is required for FPK).

3. Automatic Testing of FOL Ontologies

In this section, we describe the framework and methodology proposed in [3] for the evaluation of FOL ontologies using an existing set of conjectures consisting of falsity-tests and truth-tests.

A conjecture is decided to be entailed by the ontology only if the ATP is able to find a proof within the provided execution-time limit. The proof (which can be reported by the ATP) of a conjecture that is not-expected to be proved (i.e. a falsity-test) provides hints on the defects of the axiomatization. In general, when a proof is not found, the ATP can report, either that the conjecture is not entailed, or that the time limit was reached. In the first case, the ATP could produce a countermodel showing that the conjecture is not satisfied in a specific model of the ontology. Countermodels of truth-tests –similarly proofs of falsity-tests– could be used as hints for detecting defects in the ontology. For large ontologies, axiom selection methods [29] can be used to increase the number of useful answers (countermodels or proofs), i.e. to reduce the number of tests that exceeds the time limit. However, this approach is beyond the scope of the methodology presented in this paper. In fact, this could be a future improvement of our framework. Consequently, if ATPs find a proof, then *truth-tests* and *falsity-tests* are classified as *proved*. Otherwise, if no proof is found, then we classify both *truth-* and *falsity-tests* as *unknown* because we do not know whether the corresponding conjectures are entailed or not. For example, the truth-test *"Siblings have the same mother"* is given by the following FOL sentence:

$$\forall o1\ \forall o2\ \forall o3\ (\ (mother(o1,o2) \wedge sibling(o1,o3)\ ) \rightarrow mother(o3,o2)\ ) \tag{3}$$

and it is easily proved by ATPs to be entailed by Adimen-SUMO v2.6. Thus, the truth-test (3) is classified as *proved*. On the contrary, for the falsity-test *"Some herbivores eat animals"*, that is given by the FOL sentence:

$$\begin{aligned} \exists h\ \exists a\ \exists e\ (\quad & instance(h,Herbivore) \wedge instance(a,Animal) \\ & \wedge instance(e,Eating) \wedge agent(e,h) \wedge patient(e,a)\ ) \end{aligned} \tag{4}$$

is classified as *unknown* in Adimen-SUMO v2.6.

It is worth noting that truth-tests classified as proved will be used to grade the suitability of the axiom they come from, whereas falsity-test classified as proved will be used to detect defects in the axiomatization usually with the help of the proof reported by the ATP. As a consequence, both, the grade of suitability of a formula, and the detection on defects, rely on the ATPs utilized and also on the parameter configuration set.

In our proposal, the set of conjectures is automatically constructed by following white-box testing strategies. We create two sets[9] of conjectures: $FT(\phi)$ and $TT(\phi)$, for each axiom ϕ in the tested ontology, and this construction depends only on the syntactical form of the axiom ϕ. The purpose of $FT(\phi)$ is to detect *defects* in the axiomatization. Each conjecture α in $FT(\phi)$ is called a *falsity-test* because α is not expected to be entailed by the ontology. If some $\alpha \in FT(\phi)$ is inferred from the ontology, then ϕ contains some *redundant* subformula. For example, we detect that the following axiom, extracted from Adimen-SUMO [2], is defective:

$$\forall c \; (\; instance(c, Circle) \;\rightarrow\; \exists p \; (CenterOfCircleFn(c) = p) \;) \tag{5}$$

by means of the falsity-test:

$$\forall c \, \exists p \; (\; CenterOfCircleFn(c) = p \;) \tag{6}$$

Conjecture (6) is trivially proved since equality is reflexive. In other words, from the reflexivity axiom:

$$\forall x \; (\; x = x \;)$$

it is easy to prove that

$$\forall c \; (\; CenterOfCircleFn(c) = CenterOfCircleFn(c) \;) \tag{7}$$

and then (6) is easily entailed from (7). We conclude that the antecedent (left-hand part of the implication) of axiom (5) is redundant, i.e. the consequent (right-hand part of the implication) is entailed whatever its antecedent would be. That makes axiom (5) to be redundant itself, since by removing the redundant antecedent in (5), we get (6) and, moreover, the latter is already entailed by the ontology.
Our notion of redundancy is related to its practical use in the reasoning process, and it is formally introduced in Definition 6. However, sometimes redundancy is caused by a different kind of defect ranging from typos (e.g. a misspelled or misplaced variable symbol) to incorrect axioms (e.g. a necessary condition is not required). In particular, our proposal enables defects to be detected that we have classified in the following four classes: *typos*, *redundant axioms*, *redundant subformulas (in axioms)* and *incorrect (inaccurate) axioms*.

The set $TT(\phi)$ consists of the negation of all the conjectures in $FT(\phi)$. They are called *truth-tests* because they are expected to be inferred and they are used to grade the *suitability* of axioms for reasoning purposes. That is, whenever no conjecture in $FT(\phi)$ is classified as proved for the axiom ϕ, we identify three different grades of suitability according to the conjectures in $TT(\phi)$ that are entailed by the ontology. Axiom ϕ is *completely suitable* if the ontology entails all the truth-tests in $TT(\phi)$.

[9]They are formally defined in Definitions 2 and 4, respectively.

Otherwise ϕ is *partially suitable* if at least one conjecture in $TT(\phi)$ is proved; and ϕ is *unsuitable* if no one conjecture in $TT(\phi)$ is proved. For example, the following axiom obtained from Adimen-SUMO:

$$\forall d\ (\ instance(d,Driving)\ \rightarrow \exists v\ (\ instance(v,Vehicle) \wedge patient(d,v)\ )\ ) \qquad (8)$$

is classified as completely suitable by means of an automatically generated set of eight different truth-tests used as conjectures by the ATPs.

4. Automatic Generation of Tests

In this section, we introduce the definition of two functions FT and TT that respectively compute the sets of falsity- and truth-tests for a given axiom. These two functions are defined by (structural) induction on the syntactic structure of the input axiom, hence our automatic test generation is syntax-based.

We use lower-case Greek letters for arbitrary formulas and capital Greek letters (e.g. Φ) for (finite) sets of sentences, that equivalently can be seen as conjunctions of all their members. The notation $\phi[\alpha]$ represents the formula ϕ and, at the same time, means that α is a subformula of ϕ. We denote by $\phi[\alpha/\gamma]$ the formula that results from replacing every occurrence of α (as a subformula of ϕ) with γ. Given any FOL formula ϕ, the expressions $(\phi)^{\exists}$ and $(\phi)^{\forall}$ respectively, denote the existential and universal closure of ϕ. Note that if ϕ is a sentence, then $(\phi)^{\exists}$ and $(\phi)^{\forall}$ are identical to ϕ. Two formulas ϕ and ψ are semantically (or logically) *equivalent*, in symbols $\phi \equiv \psi$, if and only if they have exactly the same models. Given any set of sentences Φ, we say that two formulas α and β are *Φ-equivalent* if and only if $\Phi \models (\alpha \leftrightarrow \beta)^{\forall}$ (i.e. every model of Φ is also a model of $(\alpha \leftrightarrow \beta)^{\forall}$). Note that if α and β are sentences, then $(\alpha \leftrightarrow \beta)^{\forall}$ and $\alpha \leftrightarrow \beta$ are the same formula.

Most of the axioms in an ontology are (universally closed) implications, e.g. see axioms (5) and (8) above. Redundancy in the subformulas of axioms can be detected by proving unexpected conjectures. For example, an axiom $(\gamma \rightarrow \psi)^{\forall}$ is redundant in an ontology Φ, whenever Φ entails one of the conjectures $\psi^{\forall}$ or $(\neg\gamma)^{\forall}$. However, if we consider a formula $(\gamma \rightarrow (\phi \vee \psi))^{\forall}$ such that Φ entails the conjecture $(\neg\psi)^{\forall}$, then the subformula ψ is redundant in the axiom, since $(\gamma \rightarrow (\phi \vee \psi))^{\forall}$ is Φ-equivalent to $(\gamma \rightarrow \phi)^{\forall}$. Implication is an important connective in our test generation. However, for the sake of a more uniform treatment and a clearer presentation, we use the equivalence $\psi \rightarrow \gamma \equiv \neg\psi \vee \gamma$ to transform implications into disjunctions. In this sense, a subformula $\alpha \vee \beta$ can be seen as the result of transforming (and simplifying) the implication $\neg\alpha \rightarrow \beta$ or the implication $\neg\beta \rightarrow \alpha$. Consequently, given a formula $\phi[\alpha \vee \beta]$

(where the subformula $\alpha \vee \beta$ is not in the scope of negation)[10], we propose the following falsity-tests for the detection of defects by searching redundant subformulas: $\alpha^\forall$, $\beta^\forall$, $(\neg\alpha)^\forall$ and $(\neg\beta)^\forall$. Roughly speaking, when $\alpha^\forall$ is entailed by Φ, we detect that β is redundant. Symmetrically, the entailment of $(\beta)^\forall$ makes α redundant. Whenever $(\neg\alpha)^\forall$ or $(\neg\beta)^\forall$ are entailed, then α or β respectively, are redundant.

Next, we illustrate the idea for generating falsity-tests by means of an example.

Example 1. *Consider the following set of four axioms of SUMO:*

$$instance(sibling, IrreflexiveRelation) \tag{9}$$

$$domain(sibling, 1, Organism) \tag{10}$$

$$domain(sibling, 2, Organism) \tag{11}$$

$$\begin{aligned} \forall m_1\, \forall m_2\, \forall b\; (\; & (\, instance(b, Brood) \wedge member(m_1, b) \wedge member(m_2, b)) \\ & \rightarrow \quad sibling(m_1, m_2)\,) \end{aligned} \tag{12}$$

According to the type information in axioms (10-11), both arguments of sibling are restricted to be instance of Organism. Consequently, by translation (see [2] for more details), axiom (12) gives raise to the following rule-axiom in Adimen-SUMO v2.4:

$$\begin{aligned} \forall m_1\, \forall m_2\, \forall b\; (\quad (\quad & instance(m_1, Organism) \wedge \\ & instance(m_2, Organism) \wedge \\ & instance(b, Brood) \wedge \\ & member(m_1, b) \wedge \\ & member(m_2, b)\;) \rightarrow \\ sibling&(m_1, m_2)\;) \end{aligned} \tag{13}$$

Since its inner subformula is logically equivalent to the following disjunction (in negation normal form):

$$\begin{aligned} (\quad & \neg instance(m_1, Organism) \vee \\ & \neg instance(m_2, Organism) \vee \\ & \neg instance(b, Brood) \vee \\ & \neg member(m_1, b) \vee \\ & \neg member(m_2, b)\;) \qquad\qquad \vee \quad sibling(m_1, m_2) \end{aligned} \tag{14}$$

The following four conjectures belong to the set of falsity-tests for axiom (13):

[10]In the formal definition, ϕ is in negation normal form (see explanations just above Definition 2).

$$\begin{aligned}\forall m_1\, \forall m_2\, \forall b\, (\quad & \neg instance(m_1, Organism) \lor \\ & \neg instance(m_2, Organism) \lor \\ & \neg instance(b, Brood) \lor \\ & \neg member(m_1, b) \lor \neg member(m_2, b))\end{aligned} \tag{15}$$

$$\forall m_1\, \forall m_2\, (sibling(m_1, m_2)) \tag{16}$$

$$\begin{aligned}\forall m_1\, \forall m_2\, \forall b\, (\quad & instance(m_1, Organism) \land \\ & instance(m_2, Organism) \land \\ & instance(b, Brood) \land \\ & member(m_1, b) \land member(m_2, b))\end{aligned} \tag{17}$$

$$\forall m_1\, \forall m_2\, (\neg\, sibling(m_1, m_2)) \tag{18}$$

Using the methodology described in Section 3, falsity-test (15) is classified as proved, while the remaining falsity-tests (16-18) are classified as unknown. Thus, falsity-test (15) enables a defect to be detected as explained in Subsection 7.4. In fact, axiom (12) has been corrected in Adimen-SUMO v2.6.

The function FT is to be applied to every ontology axiom. Therefore, FT is defined by structural induction on a formula ϕ that belongs to the language

$$\phi ::= \ell \mid \phi \lor \phi \mid \phi \land \phi \mid \forall x \phi \mid \exists x \phi$$

where ℓ stands for literal (atom or negated atom). In addition, we can suppose that any quantifier in ϕ has a different variable symbol. Our assumption is not a limitation since it is well-known that any FOL formula can be transformed into a logically equivalent one in the above language (see e.g. [18]). The transformation follows the three initial steps of the standard algorithm that transforms any FOL formula into its conjunctive normal form:

1. Rectification or elimination of variable clashing: rename clashing variables so that each quantifier has a unique variable symbol.
2. Transformation into *arrow-free form*: repeatedly apply the following two logical equivalences left-to-right until none can be applied:
   - $\psi \leftrightarrow \gamma \equiv (\psi \to \gamma) \land (\gamma \to \psi)$
   - $\psi \to \gamma \equiv (\neg\psi) \lor \gamma$
3. Transformation into *negation normal form*: repeatedly apply the following five logical equivalences (left-to-right) until none can be applied:

- $\neg\neg\psi \equiv \psi$
- $\neg(\psi \wedge \gamma) \equiv (\neg\psi) \vee (\neg\gamma)$
- $\neg(\psi \vee \gamma) \equiv (\neg\psi) \wedge (\neg\gamma)$
- $\neg\forall x\ \psi \equiv \exists x\ \neg\psi$
- $\neg\exists x\ \psi \equiv \forall x\ \neg\psi$

In what follows, we say that the formulas in the above language are in *arrow-free and negation normal form* (in abbreviated form, *af-nnf*). For example, formula (14) in Example 1 is in af-nnf.

Definition 2. *For any af-nnf formula ϕ, the function FT is recursively defined as*

$$FT(\phi) \quad = \quad \begin{cases} \emptyset & \text{if } \phi \text{ is a literal} \\ FT_0(\phi) & \text{if } \phi = \alpha \vee \beta \\ FT(\alpha) \cup FT(\beta) & \text{if } \phi = \alpha \wedge \beta \\ FT(\alpha) & \text{if } \phi = \forall x\ \alpha \text{ or } \phi = \exists x\ \alpha \end{cases}$$

where the function FT_0 is defined as follows:

$$FT_0(\phi) \quad = \quad \begin{cases} \emptyset & \text{if } \phi \text{ is a literal} \\ \{\ (\alpha)^\forall,\ (\beta)^\forall,\ (\neg\alpha)^\forall,\ (\neg\beta)^\forall\ \} & \\ \qquad \cup FT_0(\alpha)\ \cup\ FT_0(\beta) & \text{if } \phi = \alpha \vee \beta \text{ or } \phi = \alpha \wedge \beta \\ \{\ (\alpha)^\forall,\ (\neg\alpha)^\forall\ \}\ \cup\ FT_0(\alpha) & \text{if } \phi = \forall x\ \alpha \text{ or } \phi = \exists x\ \alpha \end{cases}$$

The idea behind Definition 2 can be summed up as follows: for any falsity-tests $\{(\delta)^\forall,\ (\neg\delta)^\forall\} \subseteq FT(\phi)$, the sentence ϕ is logically equivalent to a (possibly quantified) conjunction of two formulas such that the first one is equal to a disjunction with $\overline{Qy}\ \delta$ as subformula for some prefix of quantifiers $\overline{Qy}$. Lemma 7 (in Section 6) formally states this idea and it is the key result for proving the correctness of our method (see Theorem 10 in Section 6).

Remark 3. *For the sake of simplicity, in Definition 2 we consider the binary connectives of conjunction and disjunction, however we have implemented its natural generalization to n-ary connectives. For example, for $\phi = \alpha \wedge \beta \wedge \gamma$:*

$FT_0(\phi) = \{\ (\alpha)^\forall,\ (\beta)^\forall,\ (\gamma)^\forall,\ (\neg\alpha)^\forall,\ (\neg\beta)^\forall\ (\neg\gamma)^\forall\} \cup FT_0(\alpha)\ \cup\ FT_0(\beta) \cup\ FT_0(\gamma).$

It is obvious that the function FT could produce many repeated tests. For example, for $\phi = \forall x\, ((\exists y\, \alpha) \vee (\forall z\, \beta))$ we have that

$$
\begin{aligned}
FT(\phi) &= FT(\forall x\, ((\exists y\, \alpha) \vee (\forall z\, \beta))) \\
&\overset{\forall}{=} FT_0((\exists y\, \alpha) \vee (\forall z\, \beta)) \\
&\overset{\vee}{=} \{\, (\exists y\, \alpha)^{\forall}, (\forall z\, \beta)^{\forall}, (\neg\exists y\, \alpha)^{\forall}, (\neg\forall z\, \beta)^{\forall} \,\} \cup \\
&\qquad FT_0(\exists y\, \alpha) \cup FT_0(\forall z\, \beta) \\
&\overset{\exists,\forall}{=} \{\, (\exists y\, \alpha)^{\forall}, (\forall z\, \beta)^{\forall},\ (\neg\exists y\, \alpha)^{\forall}, (\neg\forall z\, \beta)^{\forall} \,\} \cup \\
&\qquad \{\, (\alpha)^{\forall}, (\neg\alpha)^{\forall} \,\} \cup FT_0(\alpha) \cup \\
&\qquad \{\, (\beta)^{\forall}, (\neg\beta)^{\forall} \,\} \cup FT_0(\beta)
\end{aligned}
$$

where $(\forall z\, \beta)^{\forall}$ and $(\beta)^{\forall}$ are both the same formula $\forall x\, \forall z\, \beta$. In addition, $(\neg\exists y\, \alpha)^{\forall}$ and $(\neg\alpha)^{\forall}$ (using nnf) are both $\forall x\, \forall y\, \neg\alpha$. Our implementation avoids such repetitions. We provide some figures about the final amount of different falsity- and truth-tests in Section 8.

The truth-tests for an axiom ϕ are the negations of all the falsity-tests for ϕ. Hence, in the case of an axiom of the form $\phi[\alpha \vee \beta]$, we generate (among others) the truth-tests $(\neg\alpha)^{\exists}$, $(\neg\beta)^{\exists}$, $(\alpha)^{\exists}$ and $(\beta)^{\exists}$. Therefore, the function $TT(\_)$ is simply defined on the basis of $FT(\_)$ as follows.

Definition 4. *For any af-nnf formula ϕ, the function $TT(\_)$ is defined as:*

$$TT(\phi) = \{\neg\theta \mid \theta \in FT(\phi)\}$$

Example 5. *For axiom (13) in Example 1, we obtain the following truth-tests by negation of falsity-tests (15-18)*

$$
\begin{aligned}
\exists m_1\, \exists m_2\, \exists b\, (\ & instance(m_1, Organism) \wedge \\
& instance(m_2, Organism) \wedge \\
& instance(b, Brood) \wedge \\
& member(m_1, b) \wedge member(m_2, b)\ )
\end{aligned}
\tag{19}
$$

$$\exists m_1\, \exists m_2\, (\ \neg\, sibling(m_1, m_2)\ ) \tag{20}$$

$$
\begin{aligned}
\exists m_1\, \exists m_2\, \exists b\, (\ & \neg\, instance(m_1, Organism) \vee \\
& \neg\, instance(m_2, Organism) \vee \\
& \neg\, instance(b, Brood) \vee \\
& \neg\, member(m_1, b) \vee \neg\, member(m_2, b)\ )
\end{aligned}
\tag{21}
$$

$$\exists m_1\ \exists m_2\ (\ sibling(m_1,m_2)\ ) \tag{22}$$

Conjecture (19) is classified as unknown (as expected, because its corresponding falsity-test is classified as proved), while conjectures (20-22) are classified as proved, because ATPs find a proof. Since, from Example 1, we already know that axiom (13) is defective, this additional testing has no effect on the grade of suitability of axiom (13). However, in the hypothetical case that falsity-test (15) would have been classified as unknown (instead of proved), then axiom (13) would be considered partially suitable, since three truth-tests are proved, but one is unknown.

5. A Detailed Example

In this section, we provide a complete example of the sets of tests that are obtained from a formula by using the definitions introduced in Section 4.

The following three axioms are obtained from SUMO by renaming (for brevity) the original predicate and function symbols as follows: the predicates *instance* and *component* have been renamed as *i* and *c* respectively; the constants *CorpuscularObject*, *Atom*, *Proton* and *Electron* have been renamed as *CO*, *At*, *Pr* and *El* respectively; and the variables *?ATOM*, *?PROTON* and *?ELECTRON* have been renamed as *a*, *p* and *e* respectively.

$$domain(c,1,CO) \tag{23}$$

$$domain(c,2,CO) \tag{24}$$

$$\forall a\ (\ i(a,At)\ \rightarrow\ \exists p\ \exists e\ (c(p,a)\wedge c(e,a)\wedge i(p,Pr)\wedge i(e,El))\ ) \tag{25}$$

Using the type information in axioms (23-24) (see [2]), (25) is transformed into a rule axiom in Adimen-SUMO v2.4 that is logically equivalent to the following sentence in af-nnf:

$$\begin{aligned}\phi\ =\ \forall a\ (\ & \overbrace{(\neg i(a,CO))\vee(\neg i(a,At))}^{\psi_1}\ \vee \\ & \exists p\ \exists e\ (\ i(p,CO)\wedge i(e,CO)\wedge \\ & \qquad c(p,a)\wedge c(e,a)\wedge \\ & \qquad \underbrace{i(p,Pr)\wedge i(e,El)}_{\psi_2}\ )\ )\end{aligned} \tag{26}$$

Then

$$\begin{aligned}
FT(\phi) &= FT(\forall a\,(\psi_1 \vee \exists p\, \exists e\, \psi_2)) \\
&\overset{\forall}{=} FT(\psi_1 \vee \exists p\, \exists e\, \psi_2) \\
&\overset{\vee}{=} FT_0(\psi_1 \vee \exists p\, \exists e\, \psi_2) \\
&\overset{\vee}{=} \{\, (\psi_1)^\forall,\, (\exists p\, \exists e\, \psi_2)^\forall,\, (\neg\psi_1)^\forall,\, (\neg\exists p\, \exists e\, \psi_2)^\forall \,\} \cup \\
&\quad\; FT_0(\psi_1) \,\cup\, FT_0(\exists p\, \exists e\, \psi_2)
\end{aligned}$$

where $(\psi_1)^\forall$, $(\neg\psi_1)^\forall$, $(\exists p\, \exists e\, \psi_2)^\forall$ and $(\neg\exists p\, \exists e\, \psi_2)^\forall$ respectively denote the conjectures:

$$\forall a\,(\,(\neg i(a,CO)) \vee (\neg i(a,At))\,) \tag{27}$$

$$\begin{aligned}\forall a\, \exists p\, \exists e\, (\;\; & i(p,CO) \wedge i(e,CO) \wedge c(p,a) \wedge c(e,a) \wedge \\ & i(p,Pr) \wedge i(e,El)\,)\end{aligned} \tag{28}$$

$$\forall a\,(\, i(a,CO) \wedge i(a,At)\,) \tag{29}$$

$$\begin{aligned}\forall a\, \neg\, \exists p\, \exists e\, (\;\; & i(p,CO) \wedge i(e,CO) \wedge c(p,a) \wedge c(e,a) \wedge \\ & i(p,Pr) \wedge i(e,El)\,)\end{aligned} \tag{30}$$

Then, we proceed to obtain the falsity-test proposed for ψ_1:

$$\begin{aligned}
FT_0(\psi_1) &= FT_0((\neg i(a,CO)) \vee (\neg i(a,At))) \\
&\overset{\vee}{=} \{\, (\neg i(a,CO))^\forall,\, (\neg i(a,At))^\forall,\, (i(a,CO))^\forall,\, (i(a,At))^\forall \} \cup \\
&\qquad \cup FT_0(\neg i(a,CO)) \cup FT_0(\neg i(a,At)) \\
&\overset{literal}{=} \{\, (\neg i(a,CO))^\forall,\, (\neg i(a,At))^\forall,\, (i(a,CO))^\forall,\, (i(a,At))^\forall \}
\end{aligned}$$

Next, the falsity-tests for $\exists p\, \exists e\, \psi_2$ are calculated as follows:

$$\begin{aligned}
FT_0(\exists p\, \exists e\, \psi_2) &= \{\, (\exists e\, \psi_2)^\forall, (\neg\exists e\, \psi_2)^\forall \,\} \cup FT_0(\exists e\, \psi_2) \\
&\overset{\exists}{=} \{\, (\exists e\, \psi_2)^\forall, (\neg\exists e\, \psi_2)^\forall \,\} \cup \\
&\qquad\qquad \{\, (\psi_2)^\forall, (\neg(\psi_2))^\forall \,\} \cup FT_0(\psi_2)
\end{aligned}$$

Of the above tests, $(\neg\exists e\, \psi_2)^\forall$ and $(\neg\psi_2)^\forall$ are identical. Finally, the tests that are obtained from ψ_2 are (see Remark 3):

$$\begin{aligned}
FT_0(\psi_2) &= FT_0(i(p,CO) \wedge i(e,CO) \wedge c(p,a) \wedge c(e,a) \wedge i(p,Pr) \wedge i(e,El)) \\
&= \{\ (i(p,CO))^{\forall},\ (i(e,CO))^{\forall},\ c(p,a))^{\forall}, \\
&\quad (c(e,a))^{\forall},\ (i(p,Pr))^{\forall},\ (i(e,El))^{\forall}\} \cup \\
&\quad \{\ (\neg i(p,CO))^{\forall}, (\neg i(e,CO))^{\forall},\ \neg c(p,a))^{\forall}, \\
&\quad (\neg c(e,a))^{\forall}, (\neg i(p,Pr))^{\forall}, (\neg i(e,El))^{\forall}\ \}
\end{aligned}$$

Some of the tests in $FT_0(\psi_2)$ are identical up to renaming of quantified variables. For example, the tests $(i(p,CO))^{\forall}$ and $(i(e,CO))^{\forall}$ are identical *up to renaming (u.t.r.)* of the universally quantified variable. Additionally, test $(i(a,CO))^{\forall}$, which is also u.t.r. identical to both, is already in the set $FT_0(\psi_1)$. Consequently, the number of tests in $FT(\phi)$ can be substantially reduced by removing duplicates. In total, 28 falsity-tests are obtained from $FT(\phi)$, from which 13 redundant tests can be removed. It is worth noting that additional duplicated tests can arise when combining sets of falsity-tests for different axioms. To sum up, the set of falsity-tests for ϕ is:

$$\begin{aligned}
FT(\phi) &= \{\ (\psi_1)^{\forall}, (\neg\psi_1)^{\forall}, (\exists e\ \psi_2)^{\forall}, (\neg\exists e\ \psi_2)^{\forall}, \\
&\quad (\neg i(a,CO))^{\forall}, (i(a,CO))^{\forall}, (\neg i(a,At))^{\forall}, \\
&\quad (i(a,At))^{\forall}, (\psi_2)^{\forall}, (c(p,a))^{\forall}, \\
&\quad (\neg c(p,a))^{\forall}, (i(p,Pr))^{\forall}, (\neg i(p,Pr))^{\forall}, \\
&\quad (i(e,El))^{\forall}, (\neg i(e,El))^{\forall}\ \}
\end{aligned}$$

In Section 7 we report on defects found using this set of falsity-tests.

Regarding truth-tests, we obtain the following set of tests by negating the falsisty-tests in $FT(\phi)$:

$$\begin{aligned}
TT(\phi) &= \{\ (\neg\psi_1)^{\exists}, (\psi_1)^{\exists}, (\neg\exists e\ \psi_2)^{\exists}, (\exists e\ \psi_2)^{\exists}, \\
&\quad (i(a,CO))^{\exists}, (\neg i(a,CO))^{\exists}, (i(a,At))^{\exists}, \\
&\quad (\neg i(a,At))^{\exists}, (\neg\psi_2)^{\exists}, (\neg c(p,a))^{\exists}, \\
&\quad (c(p,a))^{\exists}, (\neg i(p,Pr))^{\exists}, (i(p,Pr))^{\exists}, \\
&\quad (\neg i(e,El))^{\exists}, (i(e,El))^{\exists}\ \}
\end{aligned}$$

6. Correctness

As mentioned in Section 4, our falsity-tests are related to the search of redundancies in axioms. In this section, we introduce a precise notion of redundancy and prove that our falsity-tests are relevant (or correct) in the sense that they really detect redundancy.

Definition 6. *Let Φ be a set of sentences and ϕ a sentence such that $\phi \in \Phi$. We say that ϕ contains a* redundancy *if there exists a sentence $\phi'[\alpha \vee \beta]$ that is logically equivalent to ϕ such that $\phi'[(\alpha \vee \beta)/\gamma]$ is Φ-equivalent to ϕ for $\gamma \in \{\alpha, \beta\}$. Moreover, when $\gamma = \alpha$ (resp. $\gamma = \beta$) we say that β (resp. α) is redundant in the subformula $\alpha \vee \beta$.*

In words, redundancy means that some subformula (of a sentence) can be eliminated without loss of essential information, and indeed this subformula is redundant. The sentence ϕ' mentioned in the above definition is the one provided by the following Lemma 7, i.e. the *formula associated to* ϕ by the pair of falsity-tests $\{(\delta)^\forall,\ (\neg\delta)^\forall\}$ (see Remark 8).

Lemma 7. *For any af-nnf sentence ϕ, if $\{(\delta)^\forall, (\neg\delta)^\forall\} \subseteq FT(\phi)$ then ϕ is logically equivalent to a formula of the form*

$$\overline{Qx}\ (\ ((\overline{Qy}\ \delta) \vee \gamma) \wedge \psi\ )$$

for some formulas γ and ψ and some (possibly empty) prefixes of quantifiers $\overline{Qx}$ and $\overline{Qy}$. Moreover, γ is always different from the constant false, but ψ could be the constant true.

Proof. The proof is by induction on the number of calls to $FT(\_)$ and $FT_0(\_)$ that are made to check that $\{(\delta)^\forall,\ (\neg\delta)^\forall\} \subseteq FT(\phi)$.

We are going to prove that for any subformula χ (could be a non-sentence) of ϕ, if $\{(\delta)^\forall,\ (\neg\delta)^\forall\} \subseteq FT(\chi)$ then χ is logically equivalent to some formula of the form $\overline{Qx}\ (\ ((\overline{Qy}\ \delta) \vee \gamma) \wedge \psi\ )$ and, moreover, variables $\overline{x}, \overline{y}$ are the variables in some quantifier of a subformula of χ. It is worth noting that ϕ has no variable clashing (see Section 4), hence the variables occurring in some quantifier of a subformula χ cannot appear in any quantifier of some subformula of ϕ different from χ. The base step is when $\chi = \delta \vee \gamma$ so that $FT(\phi) = FT_0(\delta \vee \gamma) = \{(\delta)^\forall,\ (\gamma)^\forall,\ (\neg\delta)^\forall,\ (\neg\gamma)^\forall\} \cup FT_0(\delta) \cup FT_0(\gamma)$. So that the property holds for empty prefixes of quantifiers and $\psi = true$. Symmetrically for $\chi = \gamma \vee \delta$.

For the inductive step, we distinguish the following cases according to the definition of $FT(\_)$ and $FT_0(\_)$:

- $FT(\chi) = FT_0(\alpha_1 \wedge \beta_1) \vee \beta) \supset FT_0(\alpha_1 \wedge \beta_1) \cup FT_0(\beta)$, and then $\{(\delta)^\forall,\ (\neg\delta)^\forall\} \supseteq FT_0(\alpha_1)$ or $\{(\delta)^\forall,\ (\neg\delta)^\forall\} \subseteq FT_0(\beta_1)$.

We prove only the case $\{(\delta)^\forall,\ (\neg\delta)^\forall\} \supseteq FT_0(\alpha_1)$, since for β_1 the proof is identical by commutativity of conjunction.

By induction hypothesis, if $\{(\delta)^\forall,\ (\neg\delta)^\forall\} \subseteq FT_0(\alpha_1)$, then there exists γ' and ψ' such that:

$$\alpha_1 \ \equiv\ \overline{Qx}\ (\ ((\overline{Qy}\ \delta) \vee \gamma') \wedge \psi'\ )$$

where variables $\bar{x},\bar{y}$ cannot appear either in β_1 or in β. Therefore:

$$\begin{aligned} \chi \ &\equiv\ (\ (\ \overline{Qx}\ (\ ((\overline{Qy}\ \delta) \vee \gamma') \wedge \psi'\ )\ ) \wedge \beta_1\ ) \vee \beta \\ &\equiv\ (\ \overline{Qx}\ (\ ((\overline{Qy}\ \delta) \vee \gamma') \wedge (\psi' \wedge \beta_1)\ )\ ) \vee \beta \\ &\equiv\ \overline{Qx}\ (\ ((\overline{Qy}\ \delta) \vee (\gamma' \vee \beta)) \wedge ((\psi' \wedge \beta_1) \vee \beta)\ ) \end{aligned}$$

Hence, we can take $\gamma' \vee \beta$ as γ and $(\psi' \wedge \beta_1) \vee \beta$ as ψ, and the property of variables $\bar{x},\bar{y}$ is trivially preserved.

- $FT(\chi) = FT_0((\forall z\, \alpha) \vee \beta) \supset FT_0(\alpha) \cup FT_0(\beta)$, and then $\{(\delta)^\forall,\ (\neg\delta)^\forall\} \subseteq FT_0(\alpha)$. By induction hypothesis, there exist γ' and ψ' such that:

$$\alpha \ \equiv\ \overline{Qx}\ (\ ((\overline{Qy}\ \delta) \vee \gamma') \wedge \psi'\ )$$

  Therefore: $\chi \ \equiv\ \forall z\ (\ \overline{Qx}\ (\ ((\overline{Qy}\ \delta) \vee \gamma') \wedge \psi'\ )\ ) \vee \beta$. Since z and $\bar{x},\bar{y}$ do not appear in β, we have that:

$$\chi \ \equiv\ \forall z\, \overline{Qx}\ (\ ((\overline{Qy}\ \delta) \vee (\gamma' \vee \beta)) \wedge (\psi' \vee \beta)\ )$$

  Thus, the property holds for an extension of the outermost prefix with z, and for $\gamma' \vee \beta$ as γ and $\psi' \vee \beta$ as ψ.

- The proof for $FT(\chi) = FT_0((\exists z\, \alpha) \vee \beta) \supset FT_0(\alpha) \cup FT_0(\beta)$ and $\{(\delta)^\forall,\ (\neg\delta)^\forall\} \subseteq FT_0(\alpha)$, is identical to the previous one.

- Suppose that $FT(\chi) = FT(\alpha \wedge \beta) = FT(\alpha) \cup FT(\beta)$ and $\{(\delta)^\forall,\ (\neg\delta)^\forall\} \subseteq FT(\alpha)$. By induction hypothesis

$$\alpha \ \equiv\ \overline{Qx}\ (\ ((\overline{Qy}\ \delta) \vee \gamma') \wedge \psi'\ )$$

  for some γ' and ψ'. Therefore, since $\bar{x}$ does not appear in β

$$\begin{aligned} \chi \ &\equiv\ (\ \overline{Qx}\ (\ ((\overline{Qy}\ \delta) \vee \gamma') \wedge \psi'\ )\ ) \wedge \beta \\ &\equiv\ \overline{Qx}\ (\ ((\overline{Qy}\ \delta) \vee \gamma') \wedge (\psi' \wedge \beta)\ ) \end{aligned}$$

  and the lemma property is true for $\gamma = \gamma'$ and $\psi = \psi' \wedge \beta$.

- For $FT(\chi) = FT(\forall z\ \alpha) \supset FT(\alpha)$ and $\{(\delta)^{\forall},\ (\neg\delta)^{\forall}\} \subseteq FT(\alpha)$. By induction hypothesis
$$\alpha \ \equiv\ \overline{Qx}\ (\ ((\overline{Qy}\ \delta) \vee \gamma) \wedge \psi\ )$$
for some γ and ψ. Therefore:
$$\chi \ \equiv\ \forall z\, \overline{Qx}\ (\ ((\overline{Qy}\ \delta) \vee \gamma) \wedge \psi\ )$$
This ensures the lemma property by enlarging the outermost prefix of quantifiers with $\forall z$.

- The proof for $FT(\chi) = FT(\exists z\ \alpha) \supset FT(\alpha)$ and $\{(\delta)^{\forall},\ (\neg\delta)^{\forall}\} \subseteq FT(\alpha)$ is identical to the previous one.

Therefore, for any $\{(\delta)^{\forall},\ (\neg\delta)^{\forall}\} \subseteq FT(\phi)$, the axiom ϕ is logically equivalent to a formula of the form $\overline{Qx}\ (\ ((\overline{Qy}\ \delta) \vee \gamma) \wedge \psi\ )$. □

Remark 8. *In what follows, we say that* $(\overline{Qy}\ \delta) \vee \gamma)$ *is the (sub)formula associated to* ϕ *by the pair of falsity-tests* $\{(\delta)^{\forall},\ (\neg\delta)^{\forall}\}$.

Next, we introduce the notion of *relevant* falsity-test in order to set out the correctness result in Theorem 10.

Definition 9. *Let* Φ *be any set of af-nnf sentences and* ϕ *any sentence such that* $\phi \in \Phi$. *If* $(\delta)^{\forall} \in FT(\phi)$ *(resp.* $(\neg\delta)^{\forall} \in FT(\phi)$*), we say that* $(\delta)^{\forall}$ *(resp.* $(\neg\delta)^{\forall}$*) is a relevant falsity-test for* Φ *whenever* $\Phi \models \delta^{\forall}$ *(resp.* $\Phi \models (\neg\delta)^{\forall}$*).*

If $\delta^{\forall}$ or $(\neg\delta)^{\forall}$ is relevant, then the redundant subformula of ϕ' is a superformula of δ and ϕ contains some redundancy. Redundant subformulas reveal the existence of defects. In addition, the proof obtained for falsity-tests can assist the correction of defects, but the correction itself is still a manual task. Some real examples on this issue are described in Section 7.

Next, we provide a formal proof of the relevance of falsity-tests.

Theorem 10. *For any consistent set of af-nnf sentences* Φ *such that* $\phi \in \Phi$, *each conjecture in* $FT(\phi)$ *is a relevant falsity-test for* Φ.

Proof. Let $\{(\delta)^{\forall},\ (\neg\delta)^{\forall}\} \subseteq FT(\phi)$. By Lemma 7, ϕ is logically equivalent to:
$$\phi' \ =\ \overline{Qx}\ (\ (\overline{Qy}\ \delta) \vee \gamma\ ) \wedge \psi\ )$$
Hence, we check the following two statements:

(a) If $\Phi \models (\delta)^{\forall}$, then $\Phi \models \phi' \leftrightarrow \phi'[(\ (\overline{Qy}\ \delta) \vee \gamma\ )/(\overline{Qy}\ \delta)]$.

(b) If $\Phi \models (\neg\delta)^{\forall}$, then $\Phi \models \phi' \leftrightarrow \phi'[(\,(\overline{Qy}\,\delta) \vee \gamma\,)/\gamma]$.

Note that each statement not only ensures that the corresponding subformula is redundant in ϕ', but also the substitution specifies what the redundant subformula is.

In order to check (a) and (b), we proceed by substitutivity of subformulas that are Φ-equivalent. For statement (a), if $\Phi \models (\delta)^{\forall}$, then it is trivial that $\Phi \models (\overline{Qy}\,\delta)^{\forall}$. Hence, $(\overline{Qy}\,\delta) \vee \gamma$ is Φ-equivalent to $\overline{Qy}\,\delta$. For statement (b), if $\Phi \models (\neg\delta)^{\forall}$, then $\Phi \models (\neg\overline{Qy}\,\delta)^{\forall}$. Hence, $(\overline{Qy}\,\delta) \vee \gamma$ is Φ-equivalent to γ. □

Theorem 10 ensures the correctness of our method. By Theorem 10, whenever a falsity-test $\alpha \in FT(\phi)$ is proved to be entailed by the ontology (where ϕ belongs to), we can ensure that there is a *relevant redundancy* in the axiom ϕ. Since truth-tests are the negations of falsity-tests, the fact of proving a truth-test guarantees the absence of a possible redundancy. Consequently, the complete suitability of an axiom ϕ in the ontology depends on proving every test in $TT(\phi)$.

According to the above definitions and results, our method relies on testing whether the whole ontology entails a set of tests. In particular, let α be a test that has been generated from an axiom ϕ, then ϕ is included in the premises used by the ATP to check whether α is entailed. From the practical point of view this is the easiest way to perform the testing, since the set of premises is fixed for testing the whole set of generated tests. From the theoretical point of view, Theorem 10 ensures that whenever α is proved, there is a relevant redundancy in the axiom ϕ.[11] Moreover, we can ensure that deleting the axiom ϕ from the premises (when checking α) would prevent some useful inferences revealing redundancies. Let us explain here a simplification of a real example to illustrate this matter.

Example 11. *Consider an ontology Φ formed by three axioms:*

$$\begin{aligned}
\phi_1 &= \forall x\,\forall y\,(\,(p(x) \wedge q(x,y)) \rightarrow r(x)\,) \qquad (31)\\
\phi_2 &= \forall x\,(\,p(x) \rightarrow \neg r(x)\,)\\
\phi_3 &= \forall x\,(\,(\exists y\, q(x,y)) \rightarrow \neg r(x)\,)
\end{aligned}$$

It is easy to see that a falsity-text in $FT(\phi_1)$ is $\forall x\,\forall y\,(\,\neg p(x) \vee \neg q(x,y)\,)$ and also that this falsity-test is entailed by Φ, but it is not entailed by $\Phi \setminus \{\phi_1\}$. Hence, the redundancy of the antecedent of (31) can be detected only if ϕ_1 belongs to the the set of premises.

[11] A different issue, that we discuss in Section 7, is to look for the defect that causes this redundancy.

The above example is an abstraction and simplification of the first example explained in Subsection 7.1, there we also explain that it is really caused by a simple typo in ϕ_1. In fact, the typo in the real example corresponds to have written $r(x)$ instead of $r(y)$ in ϕ_1 above.

7. Experimentation: Examples of Detected Defects

In this section, we illustrate with examples the defects that we have detected —using an implementation of our methodology— in the ontologies DOLCE, FPK and Adimen-SUMO (see Section 2 for an introduction to them). Our method is based on detecting redundancies. To be more precise our technique is focussed in looking for *redundant subformulas inside axioms* (using falsity-tests) or their absence (using truth-tests). Indeed, according to Theorem 10, whenever an ATP proves the entailment of a falsity-test α extracted from an axiom ϕ, there is a disjunction $\alpha \vee \beta$ that is a subformula of ϕ such that either α or β are redundant in the sense that the disjunction $\alpha \vee \beta$ can be substituted by one of its disjuncts (α or β) preserving equivalence. So, in some sense, these redundancies are the defects we are basically looking for, but there are different causes that provoke a subformula to be technically redundant in the sense of Theorem 10. On one hand, some redundancies arise because there is a different problem either in the concerned axiom or in some set of axioms that define related concepts. For example, a typo in the concerned axiom or some inaccuracy in the axioms defining a concept related with the concerned axiom. In other words, sometimes the defect can be fixed as expected (i.e. by eliminating a redundant subformula), but sometimes the detected redundancy could be caused by a different defect. For example, the redundancy detected in (31) would be (indeed, it is) really caused by a typo: the consequent of (31) should by $r(y)$ instead of $r(x)$. On the other hand, sometimes redundant axioms are intentionally introduced with different objectives, e.g. a) to facilitate the proof of conjectures; b) to maintain an uniform style of modelling; c) to communicate design decisions; d) to improve the documentation of the ontology. Additionally, automatic transformations often introduce redundant subformulas. Since the correction of defects is not automatic but manual, the ontologist/expert has to decide whether the detected redundancies need to be corrected or not. In Subsections 7.1-7.4, we provide some examples of detected redundancies that –for that reason– do not require any correction, along with other examples that require correction. For the sake of presentation we split the founded defects in four categories: *typos*, *redundant axioms*, *redundant subformulas (in axioms)* and *incorrect (inaccurate) axioms*. We provide examples of each category in the following four subsections.

7.1 *Typos*

Typos are syntactical errors that are very simple to correct. A first example of typo was in the following axiom of Adimen-SUMO:

$$\begin{aligned} \forall c\, \forall g\, (\quad & (\ instance(c, OrchestralConducting) \wedge patient(c, g)) \\ & \rightarrow \ instance(c, Orchestra) \ ) \end{aligned} \tag{32}$$

which aims to state that every participant in an instance of the process *OrchestralConducting* is an instance of *Orchestra*. However, the following falsity-test was proved:

$$\forall c\, \forall g\, (\, (\, \neg instance(c, OrchestralConducting) \vee \neg patient(c, g)\,) \tag{33}$$

This suggest that the antecedent of axiom (32) is redundant, however the proof given by the ATP utilizes an axiom stating that the first argument of *patient* (i.e. c in (32)) should be a *Process*. By disjointness of the class *Process* with the class *Orchestra* –which is entailed passing through several superclasses of the latter– it is deduced that c cannot be an instance of *Orchestra*, hence using axiom (32) itself, falsity-test (33) is entailed. The hint that the conflict comes from the type of the first argument of *patient* made us realize that it is the second (but not the first) argument of *patient* that should be an *Orchestra*. Hence, in axiom (32), the third occurrence of c is a typo. In Adimen-SUMO v2.6, it has been replaced by g:

$$\begin{aligned} \forall c\, \forall g\, (\quad & (\ instance(c, OrchestralConducting) \wedge patient(c, g)) \\ & \rightarrow \ instance(g, Orchestra) \ ) \end{aligned} \tag{34}$$

An example of a typo that was detected in KIF-DOLCE is the following axiom, where *sb* stands for the subsumption relation and *psb* stands for the proper subsumption relation:

$$\begin{aligned} \forall w\, \forall f\, \forall g\, (\quad & (\ world(w) \wedge universal(f) \wedge universal(g)\ ) \\ & \rightarrow \ (\, psb(w, f, g) \leftrightarrow (sb(w, f, g) \wedge \neg sb(w, f, g))\,)\,) \end{aligned} \tag{35}$$

The following falsity-text was generated from this axiom:

$$\forall w\, \forall x\, \forall y\, (\, \neg psb(w, x, y)\,) \tag{36}$$

It was classified as proved, and its proof reveals that, in axiom (35), the relation *psb* was incorrectly defined: the last two occurrences of f and g in axiom (35) were swapped, i.e. the last literal in (35) should be $\neg sb(w, g, f)$. After correcting this typo, 606 falsity-tests turned from proved into unknown.

7.2 *Redundant Axioms*

Redundant axioms are axioms that do not add any reasoning power to the ontology, because they are already entailed. An example is the Adimen-SUMO axiom which yields the formula (5) in Section 3. As explained there, this defect was detected by means of falsity-test (6). In Adimen-SUMO v2.6 axiom (5) is removed, though a proper axiomatization of *CenterOfCircleFn* would be more convenient.
We have also found an example of a redundant axiom in KIF-DOLCE (after we had corrected axiom (35)). The role of the following axiom is to define the disjointness relation *dj* of universals:

$$\begin{aligned} \forall w\, \forall f\, \forall g\, (\quad & (\ world(w) \wedge universal(f) \wedge universal(g)\ ) \\ & \rightarrow\ (\ dj(w,f,g) \leftrightarrow \phi\ )\ ) \end{aligned} \tag{37}$$

where ϕ is the formula that states the disjointness of universals *?F* and *?G*, but it is not relevant for the present discussion. The following falsity-test is proved:

$$\forall w\ (\ \neg world(w)\ ) \tag{38}$$

Therefore, the negation of the antecedent of axiom (37) is inferred from the ontology. Hence, the consequent of axiom (37) is redundant, and thus the axiom itself is classified as redundant. This means that *dj* has no associated definition (axiomatization) in KIF-DOLCE. By analysing the proof provided by the ATP, we discover that the axiomatization in KIF-DOLCE prevents the introduction of worlds and particulars. In fact, in KIF-DOLCE the definition of any world or any particular yields to a contradiction. This defect does not seem to be trivial to fix and it is beyond the scope of this paper. In KIF-DOLCE, there are 91 different falsity-tests that were proved due to this defect.

In the ontology FPK, the following atom:

$$p(S(Bool,T)) \tag{39}$$

encodes the boolean constant for truth as a formula, indeed it is called *aTRUTH*. Atom (39) is used as a subformula in 19 axioms, which produces 14 unique falsity-tests enabling the detection of the redundant uses of $p(S(Bool,T)$. For example, $p(S(Bool,T))$ is redundantly used in the following axiom (called *aREFLu_CLAUSE*):

$$\forall a\, \forall x\, (\ (S(a,x) = S(a,x))\ \rightarrow\ p(S(Bool,T)\ )\ ) \tag{40}$$

Indeed, for this axiom, we generate the falsity-test:

$$\forall a\, \forall x\, (\ S(a,x) = S(a,x)\ ) \tag{41}$$

which is easily proved since equality is defined as being reflexive in FOL. Consequently, axiom (40) is redundant.

7.3 *Redundant Subformulas*

Redundant subformulas (in axioms) can be simply removed from axioms without affecting the set of conjectures that can be entailed from the ontology. This kind of redundancy is often introduced by automatic transformation of axioms, such as the translation of *domain axioms* in Adimen-SUMO or the transformation of KIF-DOLCE into CASL-DOLCE. An example is given by the following axioms in CASL-DOLCE, which define the disjointness of the top classes *endurant*, *perdurant* and *quality* with *abstract*:

$$\forall y_0 \; (\; aB(y_0) \; \rightarrow \; pT(y_0) \;) \tag{42}$$

$$\forall x_1 \; (\; eDorPDorQ(x_1) \; \rightarrow \; pT(x_1)) \;) \tag{43}$$

$$\forall x_0 \; (\; pT(x_0) \; \rightarrow \; \neg(aB(x_0) \wedge eDorPDorQ(x_0)) \;) \tag{44}$$

It is easy to see that the falsity-test

$$\forall x_0 \; \neg(aB(x_0) \wedge eDorPDorQ(x_0)) \tag{45}$$

is entailed by axioms (42-44). This ensures that the antecedent of axiom (44) is redundant. Therefore, we have replaced axiom (44) with (45).

7.4 *Incorrect Axioms*

Incorrect (or inaccurate) axioms are sentences giving an inaccurate definition of the term they aim to define.
A first example of an incorrect axiom is given by axiom (12) from Adimen-SUMO. Falsity-test (15) is proved, suggesting that the antecedent of axiom (12) is redundant. The proof provided by the ATP reveals that the problem is related with axiom (9) which ensures that the relation *sibling* is irreflexive. Consequently, in Adimen-SUMO v2.6, we correct this defect by replacing axiom (12) with:

$$\begin{aligned} \forall m_1 \, \forall m_2 \, \forall b \; (\; (& instance(b, Brood) \wedge member(m_1, b) \wedge member(m_2, b) \wedge m_1 \neq m_2) \\ & \rightarrow \quad sibling(m_1, m_2) \;) \end{aligned} \tag{46}$$

Another example of an incorrect axiom is given in Section 5: falsity-test (27) is proved, and therefore the antecedent of the implication in axiom (25) seems to be redundant. However, the proof of this falsity-test reveals that the class *Atom* (abbreviated *At*) is a subclass of *Substance* and the latter does not have common instances with *CorpuscularObject* (abbr. *CO*), because *Substance* and *CO* are disjoint classes. By axiom (24), on the domain of predicate *c* (recall *component*), the variable *a* must be an instance of CO. We realized that the redundancy of the antecedent of axiom (25) is due to this misuse of the relation c (recall *component*) in the consequent of (25). In Adimen-SUMO v2.6, we fix this defect by replacing relation *component* (abbr. c) by *part* in axiom (25). The new axiom is:

$$\forall a \; (\; i(a,At) \; \rightarrow \; \exists p \, \exists e \; (part(p,a) \wedge part(e,a) \wedge i(p,Pr) \wedge i(e,El)) \;) \qquad (47)$$

Before fixing that, falsity-test (30) was also proved as *Proton* (abbr. Pr) and *Electron* (abbr. El) are also subclasses of *Substance* and, therefore, they do not have common instances with CO (*CorpuscularObject*). This proved falsity-test is related to the redundancy of the consequent of axiom (25), however the proof reveals the same misuse of the relation c (or *component*), so that the repaired axiom (47) also serves to fix this problem.

8. Experimentation Results for DOLCE, FPK and Adimen-SUMO

In this section, we report on the experimentation results we obtained by testing three FOL ontologies: DOLCE, FPK and Adimen-SUMO.

Table 2: Number of (tested) rules and (unique) falsity-tests for each ontology

	Rules		Falsity-Tests	
	Total	Tested	Total	Unique
KIF-DOLCE	257	215	10,151	2,138
CASL-DOLCE	416	378	6,637	1,266
FPK	78,225	5,753	163,751	37,698
Adimen-SUMO v2.4	2,785	1,622	27,392	7,996
Adimen-SUMO v2.6	2,799	1,629	27,499	8,010

We consider two different versions of DOLCE and Adimen-SUMO. In Table 2, we provide some general figures about the three ontologies. Given an axiom ϕ, our method generates at least one test for ϕ whenever the af-nnf of ϕ contains at least one disjunction connective. Hence, axioms that do not satisfy this condition –e.g. universally closed conjunctions of literals– are not tested by our method. In Table 2, the two columns "Rules" respectively stand for the total number of rule axioms in each ontology (also reported in Table 1) and the number of tested rules. In the columns "Falsity-tests" we provide the total number of falsity-tests generated for each ontology and the number of different falsity-tests. The number of truth-tests is equal to the number of falsity-tests by definition.

In the rest of this section, we report on the experimentation results for each ontology in the respective Subsections 8.1, 8.2 and 8.3. Additionally, in Subsection 8.3 we also report on the improvement Adimen-SUMO from v2.4 to v2.6. In our experimentation, we have used three different vesions of the theorem prover Vampire [51, 34], especificaly v2.6, v3.0 and v4.1, since they produce different results. We set an execution time limit of 600 seconds, running on an Intel® Xeon ® CPU E5-2640v3@2.60GHz with 2GB of RAM memory per processor. The five tested ontologies, the sets of tests and the program for its generation, and the execution reports are freely available at `http://adimen.si.ehu.es`.

Table (3) Experimentation results for DOLCE

	Tests		Axioms				Coverage		Time	
	#Ts.	Pr.	D.	C.S.	P.S.	U.	#Ax.	Perc.	$\leqslant$1	>120
FT	2,138	112	213	-	-	-	71	27.63%	20	89
TT	2,138	407	-	0	2	0	44	17.12%	406	1
Total	4276	519	213	0	2	0	77	29.96%	426	90

(a) KIF-DOLCE

	Tests		Axioms				Coverage		Time	
	#Ts.	Pr.	D.	C.S.	P.S.	U.	#Ax.	Perc.	$\leqslant$1	>120
FT	1,266	30	30	-	-	-	73	17.55%	30	0
TT	1,266	1,093	-	242	106	0	324	77.88%	1,075	0
Total	2,532	1,123	30	242	106	0	325	78.13%	1,105	0

(b) CASL-DOLCE

Tests.- #Ts.: Number of different tests; Pr.: Number of Proved tests.
Axioms.- Number of axioms classified as D: defective, C.S.: Completely Suitable;
P.S.: Partially Suitable; U.: Unknown.
Coverage.- #Ax.: Number of axioms that are involved in the set of proofs;
Perc.: Percentage over the total number of axioms in the ontology.
Time.- $\leqslant s$: Number of proofs that takes less than s seconds for $s \in \{1,10\}$,
>120: Number of proofs that takes more than 120 seconds.

8.1 *Testing DOLCE*

We consider two different FOL versions of DOLCE: KIF-DOLCE, which was obtained by following the translation described in [2], and CASL-DOLCE, which was obtained from the simplified translation of DOLCE into CASL [7] that is available in Hets [41].

In Table 3a, we provide the results we have obtained testing KIF-DOLCE with

2,138 different tests of each type (FT and TT). The notes below the table explain the quantities and abbreviations. We would like to point out that among the 2,138 falsity-tests, only 112 (5.24%) were proved, while 407 truth-tests were proved. The proved falsity-tests enable, at first sight, only 2 axioms (up to 215) be classified as partially suitable, whereas the remaining 213 axioms (99.07%) are classified as defective. However, according to the coverage measure, only 77 axioms (29.96% of total) are used by ATPs, of which 71 axioms are used in the proofs of falsity-tests. This low coverage is due to the fact that the addition of any constant representing a *world* or a *particular* yields to a contradiction in KIF-DOLCE, where *world* and *particular* are fundamental concepts. The small inconsistent susbset of formulas (caused by such addition) is repeatedly used by ATPs to prove the entailment of the falsity-tests. Hence, this is a defect of KIF-DOLCE, though it does not mean that most of the axioms in KIF-DOLCE are defective.

For CASL-DOLCE, we have obtained 1,266 different falsity-tests to test 378 rules axioms. The results of testing CASL-DOLCE are summarized in Table 3b. Only 30 falsity-tests (4.7%) were proved. In all cases, the detected defects consist in trivial redundancies that are introduced by the automatic translation from CASL into TPTP syntax. In particular, the translation of *dj* (disjointness of universals) artificially introduces implications where its antecedent is redundant (detected by 20 different falsity-tests). 30 axioms were classified as defective, but all of them are correct although containing trivial redundancies. In addition, 1,093 truth-tests (86.33%) were proved and 242 axioms are completely suitable (64.02% of tested axioms) and 106 axioms as partially suitable (28.04% of tested axioms). 324 different axioms are used in the proofs of those 1,093 truth-tests, which account for 77.88% of total axioms. Consequently, we do not detect any substantial defect in CASL-DOLCE and nearly two thirds of the axioms are classified as completely suitable. Note also that most of the proofs were made in less than one second. It is worth mentioning that CASL-DOLCE is a simplification of KIF-DOLCE and that they are not equivalent. For example, we cannot check whether conjecture (38) is entailed by CASL-DOLCE because the predicate *world* does not occur in it.

8.2 *Testing FPK*

The results of testing 5,753 of the FPK axioms using 37,698 different falsity-tests and the same number of (unique) truth-tests are summarized in Table 4a. Of the 37,698 falsity-tests, 581 (1.54%) were proved, enabling 610 axioms (10.60%) to be classified as defective. Fortunately, all detected defects are either redundant subformulas or redundant axioms, such as the example at the end of Subsection 7.3. The level of redundancy is not surprising since the main objective of the Flyspeck project is to exhaustively check a complex mathematical proof. We conclude that not only is the number of defects detected in FPK not high, but also that their nature is not critical. Indeed, only 336 different axioms are used in the proofs of the 581 proved falsity-tests.

Table (4) Experimentation results for FPK

	Tests		Axioms				Coverage		Time	
	#Ts.	Pr.	D.	C.S.	P.S.	U.	#Ax.	Perc.	$\leqslant$10	>120
FT	37,698	581	610	-	-	-	336	0.43%	115	443
TT	37,698	22,825	-	649	4,298	196	2,969	3.78%	7,508	12,796
Total	75,396	23,406	610	649	4,298	196	2,988	3.81%	7,623	13,239

(a) FPK

See table notes in Table 3.

With respect to suitability, 22,825 truth-tests (60.55%) are proved, enabling us to decide that 649 axioms are completely suitable (11.28% of tested axioms). Regarding the coverage measure, 2,988 formulas (3.81% from the total of 78,500 formulas) are used in proofs and 5,753 formulas are tested (7.33%). The small number of tested formulas is due to the fact that many formulas in the ontology do not contain any disjunctive connective, thus they are out of the scope of our technique. In other words, large parts of FPK remain untested due to its syntactic form. We guess that a different (syntactically-based) test generation strategy could allow a larger coverage for FPK. It is worth mentioning that the set of axioms used in the proofs of truth- and falsity-tests are disjoint. The execution times reflect that FPK encodes a complex mathematical proof. In fact, 443 of the 581 proofs of falsity-tests, and 12,796 of the 22,825 proofs of truth-tests, take more than 120 seconds.

8.3 *Testing and Improving Adimen-SUMO*

In this subsection, we describe the process of testing and improving Adimen-SUMO v2.4. Indeed, by correcting all the defects that were detected, we obtained Adimen-SUMO v2.6, which was also evaluated.

We tested 1,622 axioms (58.24% of rules) in Adimen-SUMO v2.4 utilizing a set of 7,996 different falsity-tests. The results of this experiment are summarized in Table 5a. Although no defect was found by using the set of CQs proposed in [3], most of the new falsity-tests (7,991) have been proved. Moreover, all the tested axioms were classified as defective, despite the ATPs proved the only 5 truth-tests that were built as the negation of the 5 non-proved falsity-tests). The coverage for Adimen-SUMO v2.4 is only 1.35%. This is due, likewise for KIF-DOLCE, to the fact that the proofs of the whole set of falsity-tests are based on a very small subset of axioms: 96 axioms (1.29%). We correct every defective axiom, and apply our testing methodology again to the resulting ontology, producing once more a new set of defective axioms. Consequently, we have iteratively proceeded in this way until no new defect is found. This requires 9 iterations, after which we obtained Adimen-SUMO v2.6. In total, we have corrected 21 typos, 3 redundant axioms, 17 incorrect axioms and 47 redundant subformulas (in axioms). In addition, 24 defective axioms –that were detected in preliminary experimentations– have been corrected previously to this test.

Table (5) Experimentation results for Adimen-SUMO

	Tests		Axioms				Coverage		Time	
	#Ts.	Pr.	D.	CS.	P.S.	U.	#Ax.	Perc.	$\leqslant$10	>120
FT	7,996	7,991	1,622	-	-	-	96	1.29%	2	7,955
TT	7,996	5	-	0	0	0	14	0.19%	5	0
Total	15,992	7,996	1,622	0	0	0	100	1.35%	7	7,955

(a) Adimen-SUMO v2.4

	Tests		Axioms				Coverage		Time	
	#Ts.	Pr.	D.	CS.	P.S.	U.	#Ax.	Perc.	$\leqslant$10	>120
FT	8,010	0	0	-	-	-	-	-	-	-
TT	8,010	6,698	-	881	748	0	3,830	51,53%	6372	7

(b) Adimen-SUMO v2.6

See table notes in Table 3.

In Adimen-SUMO v2.6, see Table 5b, we tested 1,629 axioms (58.20% of 2,799 rules) with a set of 8,010 different falsity-tests (and the same number of truth-tests). No falsity-test is proved, whereas 6,698 truth-tests (83.62%) are proved. The latter result allows us to classify 881 axioms as being completely suitable (54.08% of tested axioms) and 748 axioms as partially suitable (45.92%). The coverage of Adimen-SUMO v2.6 is 51.53%: 3,830 different axioms are used in the set of truth-test proofs. That coverage level is much more larger than the coverage we have ever accomplished using black-box testing techniques [4]. Thus, we can conclude that more than a half of Adimen-SUMO v2.6 has been validated using the methodology introduced in this paper.

9. Conclusions and Future Work

This work offers a new practical insight towards the automatic testing of first-order logic (FOL) ontologies using ATPs. In addition, we provide formal proofs of the correctness of the proposed tests and report on the practical application of our methodology to four different FOL ontologies: Adimen-SUMO, KIF-DOLCE, CASL-DOLCE and FPK. Using our methodology, we have been able to detect some defects in all of those ontologies, although in the last two each defect can be trivially solved. In the case of Adimen-SUMO, we have applied our white-box testing approach to Adimen-SUMO v2.4 and manually corrected all defects that have been detected. Following 9 iterations of the process of testing and correcting, we obtained Adimen-SUMO v2.6, where our implementation does not detect any defect. Moreover, 54.32% of the axioms in Adimen-SUMO v2.6 were classified as completely suitable for reasoning purposes and 83.25% of the tested axioms were classified as suitable.

It is worth highlighting that testing the suitability for reasoning purposes of axioms is often much more interesting than checking the consistency of ontologies. For example, KIF-DOLCE is proved to be consistent by Vampire v4.1 although 99% of its axioms are classified as defective and none is classified as suitable for reasoning purposes. It is also particularly interesting to evaluate the competency of axioms by following black-box testing strategies, such as the one proposed in [3].

All the resources that have been used and developed during this work are available in a single package, including:[12] a) the ontologies; b) tools for the creation of tests, the experimentation and the analysis of results; and c) the resulting tests for each ontology and the output obtained from different ATPs.

Regarding future work, ATPs are currently incorporating new techniques designed for reducing the large axiom space, such as the *axiom selection technique* [59, 39, 29, 35] and the *abstraction-refinement framework* [60, 28]. Using these techniques we hope that a part of the test that –in the experimentation reported in this paper– are unknown can be proved, hence we would find more defects in Adimen-SUMO or classify more axioms as being completely suitable. This shall also allow us to evaluate the different techniques available and possible strategies for these techniques. In addition, we could explore techniques to minimize the computational effort performed for testing based on automated reasoning algorithms for determining the impact of axioms (as in [43]) and then perform the tests in decreasing order of the impact of the axioms that generate each test.

We have introduced a correct method of white-box test generation based on finding redundancies related to the disjunction (implication) logical operator. Other methods

[12]The package is available at `http://adimen.si.ehu.es`.

for automatic generation of tests from an axiomatization can be obtained by focusing in other kinds of redundancies. Thus, we plan to study new white-box testing strategies in the future. We also plan to apply our white-box testing methodology to other ontologies and to all the future versions of Adimen-SUMO.

Finally, the proposed white-box testing opens an alternative way of checking the usefulness and non-defectiveness of axioms without proving the satisfiability of the whole ontology: in particular, by proposing tests that are specific to single axioms and by checking the use of axioms to prove the usefulness of other axioms.

Acknowledgements

The authors are grateful to the anonymous reviewers for their valuable comments that improve this publication. This work has been partially funded by the Spanish Projects TUNER (TIN2015-65308-C5-1-R), COMMAS (TIN2013-46181-C2-2-R), and GRAMM (TIN2017-86727-C2-2-R) and the Basque Projects GIU15/30 and GIU18/182.

References

[1] J. Álvez, I. Gonzalez-Dios, and G. Rigau. Cross-checking WordNet and SUMO using meronymy. In N. Calzolari, editor, *Proc. of the 11th Int. Conf. on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA), 2018.

[2] J. Álvez, P. Lucio, and G. Rigau. Adimen-SUMO: Reengineering an ontology for first-order reasoning. *Int. J. Semantic Web Inf. Syst.*, 8(4):80–116, 2012.

[3] J. Álvez, P. Lucio, and G. Rigau. Improving the competency of first-order ontologies. In Ken Barker and José Manuél Gómez-Pérez, editors, *Proc. of the 8th Int. Conf. on Knowledge Capture (K-CAP 2015)*, pages 15:1–15:8. ACM, 2015.

[4] J. Álvez, P. Lucio, and G. Rigau. Black-box testing of first-order logic ontologies using WordNet. *CoRR*, abs/1705.10217, 2017.

[5] J. Álvez, P. Lucio, and G. Rigau. Evaluating automated theorem provers using Adimen-SUMO. In L. Kovács and A. Voronkov, editors, *Proc. of the 3rd Vampire Workshop (Vampire 2016)*, volume 44 of *EPiC Series in Computing*, pages 74–82. EasyChair, 2017.

[6] J. Álvez and G. Rigau. Towards cross-checking WordNet and SUMO using meronymy. In P. Vossen, C. Fellbaum, and F. Bond, editors, *Proc. of the 9th Global WordNet Conference (GWC 2018)*, 2018.

[7] E. Astesiano, M. Bidoit, H. Kirchner, B. Krieg-Brückner, P. D. Mosses, D. Sannella, and A. Tarlecki. CASL: the Common Algebraic Specification Language. *Theoretical Computer Science*, 286(2):153–196, 2002.

[8] J. Atserias, G. Rigau, and L. Villarejo. Spanish WordNet 1.6: Porting the Spanish WordNet across Princeton versions. In *Proceedings of the 4th Int. Conf. on Language Resources and Evaluation (LREC 2004)*, 2004.

[9] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A nucleus for a web of open data. In Aberer, K. et al., editor, *The Semantic Web*, LNCS 4825, pages 722–735. Springer, 2007.

[10] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, New York, NY, USA, 2nd edition, 2010.

[11] F. Baader and R. Peñaloza. Axiom pinpointing in general tableaux. *Journal of Logic and Computation*, 20(1):5–34, 2010.

[12] F. Baader and B. Suntisrivaraporn. Debugging SNOMED CT using axiom pinpointing in the description logic EL+. In R. Cornet and K. A. Spackman, editors, *Proc. of the 3rd Int. Conf. on Knowledge Representation in Medicine ((KR-MED 2008)*, 2008.

[13] C. Bezerra, F. Freitas, and F. Santana. Evaluating ontologies with competency questions. In *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 3, pages 284–285, 2013.

[14] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. DBpedia - A crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154–165, 2009.

[15] E. Davis and G. Marcus. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Commun. ACM*, 58(9):92–103, aug 2015.

[16] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[17] M. Fernández-López, A. Gómez-Pérez, and M. C. Suárez-Figueroa. Methodological guidelines for reusing general ontologies. *Data & Knowledge Engineering*, 86:242–275, 2013.

[18] M. Fitting. *First-order Logic and Automated Theorem Proving*. Springer-Verlag New York, Inc., New York, NY, USA, 1990.

[19] G. Friedrich and K. Shchekotykhin. A general diagnosis method for ontologies. In Y. Gil, E. Motta, V. R. Benjamins, and M. A. Musen, editors, *The Semantic Web – ISWC 2005, 4th International Semantic Web Conference*, pages 232–246, Berlin, Heidelberg, 2005. Springer.

[20] A. Gangemi, N. Guarino, C. Masolo, and A. Oltramari. Sweetening WordNet with DOLCE. *AI Magazine*, 24(3):13–24, 2003.

[21] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider. Sweetening ontologies with DOLCE. In A. Gómez-Pérez et al., editor, *Knowledge Engin. and Knowledge Manag.: Ontologies and the Semantic Web*, LNCS 2473, pages 166–181. Springer, 2002.

[22] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, and Z. Wang. HermiT: An OWL 2 reasoner. *Journal of Automated Reasoning*, 53(3):245–269, Oct 2014.

[23] T. Gruber. Ontology. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems*, pages 1963–1965. Springer US, 2009.

[24] M. Grüninger and M. S. Fox. Methodology for the design and evaluation of ontologies. In *Proc. of the Workshop on Basic Ontological Issues in Knowledge Sharing (IJCAI 1995)*, 1995.

[25] N. Guarino and C. A. Welty. An overview of ontoclean. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 151–171. Springer Berlin Heidelberg, 2004.

[26] T. C. Hales and S. P. Ferguson. The Kepler conjecture. *Discrete & Computational Geometry*, 36(1):1–269, 2006.

[27] T. C. Hales, J. Harrison, S. McLaughlin, T. Nipkow, S. Obua, and R. Zumkeller. A revision of the proof of the Kepler conjecture. *Discrete & Computational Geometry*, 44(1):1–34, 2010.

[28] Julio Cesar Lopez Hernandez and Konstantin Korovin. Towards an abstraction-refinement framework for reasoning with large theories. In Thomas Eiter, David Sands, Geoff Sutcliffe, and Andrei Voronkov, editors, *IWIL Workshop and LPAR Short Presentations*, volume 1 of *Kalpa Publications in Computing*, pages 119–123. EasyChair, 2017.

[29] K. Hoder and A. Voronkov. Sine qua non for large theory reasoning. In Nikolaj Bjørner and Viorica Sofronie-Stokkermans, editors, *Automated Deduction – CADE-23*, pages 299–314, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[30] M. Horridge. *Justification based explanation in ontologies*. PhD thesis, University of Manchester, UK, 2011.

[31] Q. Ji, G. Qi, and P. Haase. A relevance-directed algorithm for finding justifications of dl entailments. In A. Gómez-Pérez, Y. Yu, and Y. Ding, editors, *The Semantic Web*, pages 306–320, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[32] A. Kalyanpur, B. Parsia, M. Horridge, and E. Sirin. Finding all justifications of OWL DL entailments. In K. Aberer, K.S. Choi, N. Noy, D. Allemang, K.I. Lee, L. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber, and P. Cudré-Mauroux, editors, *The Semantic Web 6th International Semantic Web-Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007*, volume 4825 of *Lecture Notes in Computer Science*, pages 267–280, Berlin, Heidelberg, 2007. Springer-Verlag.

[33] Knowledge Interchange Format draft proposed American National Standard (dpANS) NCITS.T2/98-004, 1998.

[34] L. Kovács and A. Voronkov. First-order theorem proving and Vampire. In N. Sharygina and H. Veith, editors, *Computer Aided Verification*, LNCS 8044, pages 1–35. Springer, 2013.

[35] E. Kuksa and T. Mossakowski. Prover-independent axiom selection for automated theorem proving in ontohub. In Pascal Fontaine, Stephan Schulz, and Josef Urban, editors, *Proceedings of the 5th Workshop on Practical Aspects of Automated Reasoning co-located with International Joint Conference on Automated Reasoning (IJCAR 2016), Coimbra, Portugal, July 2nd, 2016.*, pages 56–68, 2016.

[36] O. Kutz and T. Mossakowski. A modular consistency proof for DOLCE. In Burgard W. et al., editor, *Proc. of the 25th AAAI Conf. on Artif. Intell. (AAAI 2011)*. AAAI Press, 2011.

[37] C. Matuszek, J. Cabral, M. J. Witbrock, and J. DeOliveira. An introduction to the syntax and content of Cyc. In C. Baral, editor, *Proc. of the Spring Symposium: Formalizing and Compiling Background Knowledge and Its Appl. to Knowledge Repr. and Question Answering*, pages 44–49. AAAI Press, 2006.

[38] J. McCarthy. Artificial intelligence, logic and formalizing common sense. In R. H. Thomason, editor, *Philosophical Logic and Artificial Intelligence*, pages 161–190. Springer, 1989.

[39] J. Meng and L. C. Paulson. Lightweight relevance filtering for machine-generated resolution problems. *Journal of Applied Logic*, 7(1):41 – 57, 2009. Special Issue: Empirically Successful Computerized Reasoning.

[40] M. Minsky. *The Emotion Machine: Commonsense Thinking, Artificial Intelligence, and the Future of the Human Mind.* Simon & Schuster, 2007.

[41] T. Mossakowski, C. Maeder, and K. Lüttich. The heterogeneous tool set, Hets. In O. Grumberg and M. Huth, editors, *Proc. of the 13th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2007)*, LNCS 4424, pages 519–522. Springer, 2007.

[42] G. J. Myers, C. Sandler, and T. Badgett. *The art of software testing*. John Wiley & Sons, Inc., Hoboken, N.J., 2012.

[43] N. Nikitina, S. Rudolph, and B. Glimm. Interactive ontology revision. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 12-13:118–130, 2012.

[44] I. Niles and A. Pease. Towards a standard upper ontology. In Guarino N. et al., editor, *Proc. of the 2nd Int. Conf. on Formal Ontology in Information Systems (FOIS 2001)*, pages 2–9. ACM, 2001.

[45] I. Niles and A. Pease. Linking lexicons and ontologies: Mapping WordNet to the Suggested Upper Merged Ontology. In H. R. Arabnia, editor, *Proc. of the IEEE Int. Conf. on Inf. and Knowledge Engin. (IKE 2003)*, volume 2, pages 412–416. CSREA Press, 2003.

[46] N. F. Noy and D. L. McGuinness. Ontology development 101: A guide to creating your first ontology. Technical Report KSL-01-05 and SMI-2001-0880, Stanford Knowledge Systems Laboratory and Stanford Medical Informatics, 2001.

[47] B. Parsia, E. Sirin, and A. Kalyanpur. Debugging owl ontologies. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 633–640, New York, NY, USA, 2005. ACM.

[48] H. Paulheim and A. Gangemi. Serving dbpedia with dolce – more than just adding a cherry on top. In M. Arenas, O. Corcho, E.Simperl, M.Strohmaier, M. d'Aquin, K. Srinivas, P. Groth, M. Dumontier, J. Heflin, K. Thirunarayan, and Steffen S. Staab, editors, *The Semantic Web - ISWC 2015*, pages 180–196, Cham, 2015. Springer International Publishing.

[49] A. Pease. Standard Upper Ontology Knowledge Interchange Format. Retrieved June 18, 2009, from `http://sigmakee.cvs.sourceforge.net/sigmakee/sigma/suo-kif.pdf`, 2009.

[50] A. Pease and C. Benzmüller. Sigma: An integrated development environment for formal ontology. *AI Communications (Special Issue on Intelligent Engineering Techniques for Knowledge Bases)*, 26(1):79–97, 2013.

[51] A. Riazanov and A. Voronkov. The design and implementation of Vampire. *AI Communications*, 15(2-3):91–110, 2002.

[52] S. Schlobach, Z. Huang, R. Cornet, and F. van Harmelen. Debugging incoherent terminologies. *Journal of Automated Reasoning*, 39(3):317–349, Oct 2007.

[53] S. Schulz, G. Sutcliffe, J. Urban, and A. Pease. Detecting inconsistencies in large first-order knowledge bases. In *Proceedings of the 26th International Conference on Automated Deduction (CADE 26)*, volume 10395 of *Lecture Notes in Computer Science*, pages 310–325. Springer, 2017.

[54] K. Shchekotykhin, G. Friedrich, P. Fleiss, and P. Rodler. Interactive ontology debugging: two query strategies for efficient fault localization. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 12(0), 2012.

[55] E. Sirin, B. Parsia, B. Cuenca-Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):51–53, 2007.

[56] S. Staab and R. Studer. *Handbook on Ontologies*. Springer Publishing Company, Incorporated, 2^{nd} edition, 2009.

[57] G. Sutcliffe. The TPTP problem library and associated infrastructure. *J. Automated Reasoning*, 43(4):337–362, 2009.

[58] G. Sutcliffe. The CADE ATP system competition - CASC. *AI Magazine*, 37(2):99–101, 2016.

[59] G. Sutcliffe and A. Dvorský. Proving harder theorems by axiom reduction. In Ingrid Russell and Susan M. Haller, editors, *Proceedings of the Sixteenth International Florida Artificial Intelligence Research Society Conference, May 12-14, 2003, St. Augustine, Florida, USA*, pages 108–113, 2003.

[60] A. Teucke and C. Weidenbach. First-order logic theorem proving and model building via approximation and instantiation. In Carsten Lutz and Silvio Ranise, editors, *Frontiers of Combining Systems*, pages 85–100, Cham, 2015. Springer International Publishing.

[61] M. Teymourlouie, A. Zaeri, M. Nematbakhsh, M. Thimm, and S. Staab. Detecting hidden errors in an ontology using contextual knowledge. *Expert Systems with Applications*, 95:312–323, 2018.

[62] D. Tsarkov and I. Horrocks. FaCT++ description logic reasoner: system description. In Ulrich Furbach and Natarajan Shankar, editors, *Proc. of the 3rd International Joint Conference on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Computer Science*, pages 292–297. Springer Berlin Heidelberg, 2006.

[63] P. Vossen, E. Agirre, G. Rigau, and A. Soroa. Kyoto: A knowledge-rich approach to the interoperable mining of events from text. In A. Oltramari, P. Vossen, L. Qin, and E. Hovy, editors, *New Trends of Research in Ontologies and Lexical Resources: Ideas, Projects, Systems*, pages 65–90, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[64] Y. Zhang, D. Ouyang, and Y. Ye. Glass-box debugging algorithm based on unsatisfiable dependent paths. *IEEE Access*, 5:18725–18736, 2017.