

SAT-Based Axiom Pinpointing Revisited

Birte Glimm¹, Yevgeny Kazakov¹

¹Ulm University, Germany

Abstract

Propositional SAT solvers have been a popular way of computing justifications for ontological entailment-minimal subsets of axioms of the ontologies that entail a given conclusion. Most SAT encodings proposed for Description Logics (DLs), translate the inferences obtained by a consequence-based procedure to propositional Horn clauses, using which entailments from subsets of axioms can be effectively checked, and use modified SAT solvers to systematically search over these subsets. To avoid repeated discovery of subsets with already checked entailment, the modified SAT solvers add special blocking clauses that prevent generating truth assignments corresponding to these subsets, the number of which can be exponential, even if the number of justifications is small. In this paper, we propose alternative SAT encodings that avoid generation of unnecessary blocking clauses. Unlike the previous methods, the inferences are used not only for checking entailment from subsets of axioms, but also, as a part of the encoding, to ensure that the SAT solver generates truth assignments corresponding only to justifications.

Keywords


Description Logics, explanation, propositional satisfiability


1. Introduction


Most Description Logic Reasoners, such as CEL [1], ELK [2], FacT++ [3], HermiT [4], Konclude [5], and Pellet [6], can answer yes/no questions about ontological entailment, but very few reasoners can *explain* why the entailment holds or does not hold. Axiom pinpointing methods [7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18] try to explain entailments by determining (some or all) minimal subsets of axioms in the ontology that cause the entailment—the so-called *justifications* or *minimal axiom sets* (MinAs). A dual notion to justification is a *repair* or *minimal correction set* (MCS)—a minimal set of axioms the removal of which breaks the entailment.


One usually distinguishes between *black-box* and *glass-box* axiom pinpointing procedures. Black-box procedures use only the yes/no answers provided by the reasoner, whereas glass-box procedures also use other information, such as the set of inferences applied during the entailment test. The set of inferences can be generated using consequence-based procedures [19, 20, 21, 22] implemented by reasoners, such as CEL [1], CB [20], ELK [2], ConDOR [21], or Sequoia [23]. Using a set of inferences, for example, one can quickly test entailments from subsets of the ontology without using the reasoner. One of the popular approaches is to encode inferences as propositional Horn clauses and use propositional satisfiability (SAT) solvers to


 DL 2022: 35th International Workshop on Description Logics, August 7–10, 2022, Haifa, Israel

 birte.glimm@uni-ulm.de (B. Glimm); yevgeny.kazakov@uni-ulm.de (Y. Kazakov)

 <https://www.uni-ulm.de/in/ki/glimm/> (B. Glimm); <https://www.uni-ulm.de/in/ki/kazakov/> (Y. Kazakov)

 0000-0002-6331-4176 (B. Glimm); 0000-0002-3011-8646 (Y. Kazakov)

 © 2022 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

test entailments from candidate subsets [9, 14, 15, 18]. To avoid generation of subsets that have been already tested (or for which the answer already follows from previous tests), the methods add *blocking clauses*, which force the SAT solver to generate new models. However, in general, the number of blocking clauses can be exponential in the number of justifications. In this paper, we propose alternative SAT encodings that use the inferences to avoid generation of models (and hence of blocking clauses) that are not relevant for computing justifications or repairs.

2. Preliminaries

We assume the standard syntax and semantics of Description Logics (DLs) such as \mathcal{ALC} (see, e.g., [24]). Suppose that \models is an entailment relation between an ontology \mathcal{O} and an axiom α . A *justification* for the entailment $\mathcal{O} \models \alpha$ is a minimal subset $M \subseteq \mathcal{O}$ such that $M \models \alpha$. A *repair* for $\mathcal{O} \models \alpha$ is a minimal subset $M \subseteq \mathcal{O}$ such that $\mathcal{O} \setminus M \not\models \alpha$.

Example 1. Consider the ontology $\mathcal{O} = \{A \sqsubseteq B, B \sqsubseteq C, A \sqsubseteq D, B \sqcap C \sqsubseteq \perp\}$ and $\alpha = A \sqsubseteq C \sqcap D$. The entailment $\mathcal{O} \models \alpha$ has 2 justifications: $j_1 = \{A \sqsubseteq B, B \sqsubseteq C, A \sqsubseteq D\}$ and $j_2 = \{A \sqsubseteq B, B \sqsubseteq C, B \sqcap C \sqsubseteq \perp\}$, and 3 repairs: $r_1 = \{A \sqsubseteq B\}$, $r_2 = \{B \sqsubseteq C\}$, and $r_3 = \{A \sqsubseteq D, B \sqcap C \sqsubseteq \perp\}$.

An *inference* is an expression inf of the form $\langle \alpha_1, \dots, \alpha_n \vdash \alpha \rangle$ where $\alpha_1, \dots, \alpha_n$ is a (possibly empty) sequence of axioms called the *premises* of inf , and α is an axiom called the *conclusion* of inf . An inference $\langle \alpha_1, \dots, \alpha_n \vdash \alpha \rangle$ is *sound* if $\{\alpha_1, \dots, \alpha_n\} \models \alpha$.

Let \mathcal{I} be a set of inferences. An \mathcal{I} -*derivation* from \mathcal{O} is a sequence of inferences $d = \langle \text{inf}_1, \dots, \text{inf}_k \rangle$ from \mathcal{I} such that for every i with $1 \leq i \leq k$, and each premise α of inf_i that is not in \mathcal{O} , there exists $j < i$ such that α is the conclusion of inf_j . An axiom α is *derivable* from \mathcal{O} using \mathcal{I} (notation: $\mathcal{O} \vdash_{\mathcal{I}} \alpha$) if either $\alpha \in \mathcal{O}$ or there exists an \mathcal{I} -derivation $d = \langle \text{inf}_1, \dots, \text{inf}_k \rangle$ from \mathcal{O} such that α is the conclusion of inf_k . A set of inferences \mathcal{I} is *complete* for the entailment $\mathcal{O} \models \alpha$ if $M \models \alpha$ implies $M \vdash_{\mathcal{I}} \alpha$ for every subset $M \subseteq \mathcal{O}$.

Example 2 (Example 1 continued). Consider a set \mathcal{I} consisting of the following inferences:

$$\begin{array}{ll} \text{inf}_1: \langle A \sqsubseteq B, B \sqsubseteq C \vdash A \sqsubseteq C \rangle, & \text{inf}_4: \langle A \sqsubseteq B \sqcap C, B \sqcap C \sqsubseteq \perp \vdash A \sqsubseteq \perp \rangle, \\ \text{inf}_2: \langle A \sqsubseteq C, A \sqsubseteq D \vdash A \sqsubseteq C \sqcap D \rangle, & \text{inf}_5: \langle \vdash \perp \sqsubseteq C \sqcap D \rangle, \\ \text{inf}_3: \langle A \sqsubseteq B, A \sqsubseteq C \vdash A \sqsubseteq B \sqcap C \rangle, & \text{inf}_6: \langle A \sqsubseteq \perp, \perp \sqsubseteq C \sqcap D \vdash A \sqsubseteq C \sqcap D \rangle. \end{array}$$

Inference inf_5 has no premises; all other inferences have 2 premises. Clearly, all inferences are sound. For axiom α , and justifications j_1 and j_2 from Example 1, we have $j_1 \vdash_{\mathcal{I}} \alpha$ and $j_2 \vdash_{\mathcal{I}} \alpha$ due to the derivation $d_1 = \langle \text{inf}_1, \text{inf}_2 \rangle$ and $d_2 = \langle \text{inf}_1, \text{inf}_3, \text{inf}_4, \text{inf}_5, \text{inf}_6 \rangle$ respectively. Therefore, \mathcal{I} is complete for the entailment $\mathcal{O} \models \alpha$.

Note that if $\mathcal{O} \not\models \alpha$ then the entailment $\mathcal{O} \models \alpha$ has no justification, has a single repair $r = \emptyset$ and any set of inferences \mathcal{I} is complete for $\mathcal{O} \models \alpha$.

3. SAT-Based Axiom Pinpointing

In this section, we describe the MARCO algorithm for computing justifications and repairs [25]. This algorithm can be regarded as a common basis of most SAT-based axiom pinpointing tools such as EL+SAT [9, 26], EL2MUS [14], SATPin [15, 18], and BEACON [27]. Although these methods are frequently regarded as *glass-box* procedures, the additional information provided by the reasoner is used only to speed up the entailment tests performed by the MARCO algorithm.

Given an ontology \mathcal{O} and an axiom α the algorithm computes the set of all justifications J and minimal repairs R for the entailment $\mathcal{O} \models \alpha$ (which may or may not hold). The basic idea is as follows. Each found justification $j \in J$ determines subsets $M \subseteq \mathcal{O}$ of the ontology such that $M \models \alpha$, namely all $M \subseteq \mathcal{O}$ such that $j \subseteq M$. Similarly, each found repair $r \in R$ determines subsets $M \subseteq \mathcal{O}$ such that $M \not\models \alpha$, namely those $M \subseteq \mathcal{O}$ such that $M \cap r = \emptyset$. The SAT solver is used to discover the remaining subsets $M \subseteq \mathcal{O}$, namely those $M \subseteq \mathcal{O}$ such that (1) $j \not\subseteq M$ for every $j \in J$ and (2) $M \cap r \neq \emptyset$ for every $r \in R$. As we will see, using each such subset M either a new justification or a new minimal repair can be found.

To find a subset $M \subseteq \mathcal{O}$ for which the entailment $M \models \alpha$ is not yet known, Conditions (1) and (2) are encoded using propositional formulas. Specifically, each axiom $\alpha \in \mathcal{O}$ is assigned to a distinguished propositional variable p_α . Then each propositional assignment \mathcal{I} of these variables describes a subset $M(\mathcal{I}) = \{\alpha \in \mathcal{O} \mid p_\alpha^\mathcal{I} = 1\}$. The goal, therefore, is to construct a formula F such that for each model \mathcal{I} of F the subset $M(\mathcal{I})$ satisfies (1) and (2).

To encode Condition (1), for each $j \in J$ we add a new *blocking clause* $\bigvee \{\neg p_\beta \mid \beta \in j\}$ to F . This clause ensures that if $F^\mathcal{I} = 1$ then $p_\beta^\mathcal{I} = 0$ for some $\beta \in j$, hence $\beta \notin M(\mathcal{I})$, and so $j \not\subseteq M(\mathcal{I})$. Similarly, to encode Condition (2), for each $r \in R$, we add a new *blocking clause* $\bigvee \{p_\beta \mid \beta \in r\}$. This clause ensures that $\beta \in M(\mathcal{I})$ for some $\beta \in r$. Hence $M(\mathcal{I}) \cap r \neq \emptyset$.

The construction of the resulting formula F as well as of new justifications and repairs is described by Algorithm 1. We start with the empty set of justifications J and repairs R , as well as a tautological formula F whose models \mathcal{I} describe all subsets $M(\mathcal{I}) \subseteq \mathcal{O}$ (Line 1). Then we repeatedly search for models of this formula using a SAT solver (Lines 2–12). For each such model \mathcal{I} (Line 3), we extract the corresponding subset $M = M(\mathcal{I})$ (Line 4) and test the entailment $M \models \alpha$, e.g., using a reasoner (Line 5). If the entailment holds, this means that there exists a minimal subset $j \subseteq M$ such that $j \models \alpha$, i.e., a justification. This subset is found using function $\text{Minimize}(M \models \alpha)$ (Line 6) to be discussed next. Due to Condition (1), j must be different from any justifications in J found so far. Hence, we add this justification to J (Line 7) and add to F the encoding of (1) for this justification (Line 8). If $M \not\models \alpha$, we can compute a maximal superset $M' \supseteq M$ of M such that $M' \not\models \alpha$, and, consequently, a repair $r = \mathcal{O} \setminus M'$ (Line 10). Again, due to Condition (2), r must be different from any repairs from R found so far. Hence, we add r to R (Line 11) and add its encoding to F (Line 12).

Minimizing entailments and maximizing non-entailments are preformed using two functions $\text{Minimize}(M \models \alpha)$ and $\text{Maximize}(M \not\models \alpha)$ shown in Algorithm 1. For minimizing $M \models \alpha$, we repeatedly *remove* from M axioms $\beta \in M$ unless this *breaks* the entailment. For maximizing $M \models \alpha$, we repeatedly *add* remaining axioms $\beta \in \mathcal{O} \setminus M$ unless this *causes* the entailment. Note that the results depend on the order in which the axioms β are iterated, however, correctness of Algorithm 1 does not depend on the *choice* of minimal subsets or maximal supersets of M .

Algorithm 1: Computing all justifications and minimal repairs using a SAT solver

SAT-Pinpointing(\mathcal{O}, α):

input : ontology \mathcal{O} and axiom α

output : the set J of all justifications and R of all minimal repairs for $\mathcal{O} \models \alpha$

```

1  $J \leftarrow \emptyset, R \leftarrow \emptyset, F \leftarrow \top$ ;
2 while  $\{\mathcal{I} \mid F^{\mathcal{I}} = 1\} \neq \emptyset$  do
3    $\mathcal{I} \leftarrow \mathcal{I} : F^{\mathcal{I}} = 1$ ;
4    $M \leftarrow \{\beta \in \mathcal{O} \mid p_{\beta}^{\mathcal{I}} = 1\}$ ;
5   if  $M \models \alpha$  then
6      $j \leftarrow \text{Minimize}(M \models \alpha)$ ;
7      $J \leftarrow J \cup \{j\}$ ;
8      $F \leftarrow F \wedge \bigvee \{\neg p_{\beta} \mid \beta \in j\}$ ;
9   else
10     $r \leftarrow \mathcal{O} \setminus \text{Maximize}(M \not\models \alpha)$ ;
11     $R \leftarrow R \cup \{r\}$ ;
12     $F \leftarrow F \wedge \bigvee \{p_{\beta} \mid \beta \in r\}$ ;
13 return  $J, R$ ;

    Minimize( $M \models \alpha$ ):
14 for  $\beta \in M$  do
15   if  $M \setminus \{\beta\} \models \alpha$  then
16      $M \leftarrow M \setminus \{\beta\}$ ;
17 return  $M$ ;

    Maximize( $M \not\models \alpha$ ):
18 for  $\beta \in \mathcal{O} \setminus M$  do
19   if  $M \cup \{\beta\} \not\models \alpha$  then
20      $M \leftarrow M \cup \{\beta\}$ ;
21 return  $M$ ;

```

Example 3. Let us see how Algorithm 1 computes justifications j_1, j_2 and repairs r_1, r_2, r_3 for the entailment $\mathcal{O} \models \alpha$ in Example 1. We start by assigning propositional variables to axioms in \mathcal{O} as follows: $A \sqsubseteq B \rightsquigarrow p_1, B \sqsubseteq C \rightsquigarrow p_2, A \sqsubseteq D \rightsquigarrow p_3, B \sqcap C \sqsubseteq \perp \rightsquigarrow p_4$. Assume that each iteration of the while loop (Lines 2–12) returns a model shown in the first column of the table:

$p_1^{\mathcal{I}}$	$p_2^{\mathcal{I}}$	$p_3^{\mathcal{I}}$	$p_4^{\mathcal{I}}$	$\min(M) \models \alpha / \max(M) \not\models \alpha$	just/rep	new clause
0	0	0	0	$\emptyset \cup \{A \sqsubseteq B, B \sqsubseteq C\} \not\models \alpha$	r_3	$p_3 \vee p_4$
0	0	1	0	$\{A \sqsubseteq D\} \cup \{A \sqsubseteq B, B \sqcap C \sqsubseteq \perp\} \not\models \alpha$	r_2	p_2
0	1	1	0	$\{B \sqsubseteq C, A \sqsubseteq D\} \cup \{B \sqcap C \sqsubseteq \perp\} \not\models \alpha$	r_1	p_1
1	1	1	0	$\{A \sqsubseteq B, B \sqsubseteq C, A \sqsubseteq D\} \setminus \emptyset \models \alpha$	j_1	$\neg p_1 \vee \neg p_2 \vee \neg p_3$
1	1	0	1	$\{A \sqsubseteq B, B \sqsubseteq C, B \sqcap C \sqsubseteq \perp\} \setminus \emptyset \models \alpha$	j_2	$\neg p_1 \vee \neg p_2 \vee \neg p_4$

The second column shows the resulting set M and its minimization (if $M \models \alpha$) or maximization (if $M \not\models \alpha$). For example, the model in the first row corresponds to $M = \emptyset \not\models \alpha$, which is maximized by adding axioms $A \sqsubseteq B$ and $B \sqsubseteq C$. The third column shows the corresponding justification or repair from Example 1. The fourth column shows the clause resulting by encoding Conditions (1) and (2) for the found justification or repair. Each next propositional assignment in the first column must be a model of all previous clauses. After adding the 5 clauses in the table, the formula F becomes unsatisfiable and Algorithm 1 terminates returning the content of the third column.

An advantage of Algorithm 1 is that it can compute justifications as well as repairs. However, in applications when only the computation of justifications is of main interest, e.g., for ontology debugging, the additional overhead for computing repairs can be significant. Indeed, as next example shows, the number of repairs can be exponential in the number of justifications.

Example 4. Consider the ontology $\mathcal{O} = \{A \sqsubseteq B_i, B_i \sqsubseteq C \mid 1 \leq i \leq n\}$. Clearly, each subset $j_i = \{A \sqsubseteq B_i, B_i \sqsubseteq C\}$ ($1 \leq i \leq n$) is a justification for the entailment $\mathcal{O} \models A \sqsubseteq C$, and each subset containing exactly one axiom from each j_i ($1 \leq i \leq n$) is a repair. To compute these repairs, Algorithm 1 performs 2^n calls to the SAT solver, each resulting in a new blocking clause.

Note that any black-box procedure that computes *all* justifications must perform an exponential number of entailment tests for the ontology \mathcal{O} in Example 4 because an entailment test must be performed for the complement of *every* repair to rule out further justifications. As seen from Example 4, Algorithm 1 is particularly problematic if many, mostly independent, subsets of the ontology are responsible for the entailment. A similar example can be constructed when there are exponentially-many justifications but only linearly-many repairs (see, e.g., Example 23 in [28]). Next, we describe alternative SAT encodings using which only justifications or only repairs can be computed. Unlike Algorithm 1, the encodings are *truly* glass-box since inferences are used not only for checking entailments, but also, as a part of the encoding, to ensure that the SAT solver returns models corresponding to only justifications or only repairs.

4. Computing Just Repairs

Assume that we have a set of sound inferences \mathcal{I} that is complete for the entailment $\mathcal{O} \models \alpha$. That is, $M \vdash_1 \alpha$ iff $M \models \alpha$ for every $M \subseteq \mathcal{O}$ (see Section 2). To find a repair for $\mathcal{O} \models \alpha$, we need to find a subset $M \subseteq \mathcal{O}$ such that $M \not\vdash_1 \alpha$, i.e., α is not derivable from M using the inferences in \mathcal{I} . According to the definition of derivability, this means that (1) $\alpha \notin M$, and (2) for *every* inference $\langle \beta_1, \dots, \beta_n \vdash \alpha \rangle \in \mathcal{I}$, *some* of the premises β_i is not derivable as well. This simple observation is a basis of our new SAT encoding for computing repairs.

Let us assign to each axiom β appearing in \mathcal{I} (as a premise or a conclusion) a distinct propositional variable b_β . Intuitively, each propositional interpretation \mathcal{I} describes axioms β with $b_\beta^\mathcal{I} = 1$ that should *not* be derived from M . We call these axioms *broken* (hence the variable name b_β). According to Condition (1) above, we should have $\beta \notin M$ for all broken β , so we take $M = M(\mathcal{I}) = \{\beta \in \mathcal{O} \mid b_\beta^\mathcal{I} = 0\}$ to be all non-broken axioms in \mathcal{O} . According to Condition (2), if $\langle \beta_1, \dots, \beta_n \vdash \beta \rangle \in \mathcal{I}$ and β is broken then one of the premises β_i must be broken as well ($1 \leq i \leq n$). We can encode this condition using a clause $\neg b_\beta \vee b_{\beta_1} \vee \dots \vee b_{\beta_n}$. Let F be the conjunction of all such clauses for inferences in \mathcal{I} plus b_α . Then for every model \mathcal{I} of F , we have $M = M(\mathcal{I}) \not\vdash_1 \alpha$, which implies $M \not\models \alpha$ since \mathcal{I} is complete for $\mathcal{O} \models \alpha$. Hence we can extract a repair of $\mathcal{O} \models \alpha$ by maximizing the non-entailment $M \not\models \alpha$ just like in Algorithm 1.

Algorithm 2 describes the above encoding and computations. We start with the empty set of repairs R and initialize the formula F as conjunction of b_α and clauses for inferences in \mathcal{I} as described above (Line 1). So long F is satisfiable (Lines 2–7), we take a model of F (Line 3), extract the subset $M = M(\mathcal{I})$ of non-broken axioms in \mathcal{O} (Line 4) for which our encoding guarantees that $M \not\models \alpha$. Then we maximize this non-entailment and extract the resulting repair just like in Algorithm 1 (Line 5), add this repair to R (Line 6), and add the corresponding blocking clause (Line 7) to F to ensure that this repair is not found by subsequent models.

Example 5. Let us see how Algorithm 2 computes the repairs r_1, r_2 , and r_3 for the entailment $\mathcal{O} \models \alpha$ in Example 1 using the set of 6 (sound and complete) inferences \mathcal{I} from Example 2. Given these parameters, the function $\text{InitRepairs}(\alpha, \mathcal{I})$ creates a conjunction F of the following clauses:

Algorithm 2: Computing all repairs using a SAT solver

SAT-Repair($\mathcal{O}, \alpha, \mathsf{l}$):

input : ontology \mathcal{O} , α an axiom, l a set of sound inferences complete for $\mathcal{O} \models \alpha$
output : the set R of all repairs for $\mathcal{O} \models \alpha$

```

1  $R \leftarrow \emptyset$ ,  $F \leftarrow \text{InitRepairs}(\alpha, \mathsf{l})$ ;
2 while  $\{\mathcal{I} \mid F^{\mathcal{I}} = 1\} \neq \emptyset$  do
3    $\mathcal{I} \leftarrow \mathcal{I} : F^{\mathcal{I}} = 1$ ;
4    $M \leftarrow \{\beta \in \mathcal{O} \mid b_{\beta}^{\mathcal{I}} = 0\}$ ;
5    $r \leftarrow \mathcal{O} \setminus \text{Maximize}(M \not\models \alpha)$ ;
6    $R \leftarrow R \cup \{r\}$ ;
7    $F \leftarrow F \wedge \bigvee \{\neg b_{\beta} \mid \beta \in r\}$ ;
8 return  $R$ ;

```

InitRepairs(α, l):

```

9  $F \leftarrow b_{\alpha}$ ;
10 foreach  $\langle \beta_1, \dots, \beta_n \vdash \beta \rangle \in \mathsf{l}$  do
11    $F \leftarrow F \wedge (\neg b_{\beta} \vee \bigvee \{b_{\beta_i} \mid 1 \leq i \leq n\})$ 
12 return  $F$ ;

```

- | | | |
|-----------------------------------|---|--|
| 1. $b_{A \sqsubseteq C \sqcap D}$ | 2. $\neg b_{A \sqsubseteq C} \vee \mathbf{b}_{A \sqsubseteq B} \vee \mathbf{b}_{B \sqsubseteq C}$ | 5. $\neg b_{A \sqsubseteq \perp} \vee b_{A \sqsubseteq B \sqcap C} \vee \mathbf{b}_{B \sqcap C \sqsubseteq \perp}$ |
| | 3. $\neg b_{A \sqsubseteq C \sqcap D} \vee b_{A \sqsubseteq C} \vee \mathbf{b}_{A \sqsubseteq D}$ | 6. $\neg b_{\perp \sqsubseteq C \sqcap D}$ |
| | 4. $\neg b_{A \sqsubseteq B \sqcap C} \vee \mathbf{b}_{A \sqsubseteq B} \vee b_{A \sqsubseteq C}$ | 7. $\neg b_{A \sqsubseteq C \sqcap D} \vee b_{A \sqsubseteq \perp} \vee b_{\perp \sqsubseteq C \sqcap D}$ |

We highlighted in bold the variables assigned to axioms in \mathcal{O} since propositional assignments of these variables determine the subsets M and the resulting repairs. The table below shows examples of models of F and the resulting repairs obtained at each iteration of Algorithm 2:

$b_{\beta}^{\mathcal{I}} = 1$	<i>repair</i>	<i>new clause</i>
$b_{A \sqsubseteq C \sqcap D}$ $b_{A \sqsubseteq C}$ $\mathbf{b}_{A \sqsubseteq B}$ $b_{A \sqsubseteq \perp}$ $b_{A \sqsubseteq B \sqcap C}$	r_1	$\neg \mathbf{b}_{A \sqsubseteq B}$
$b_{A \sqsubseteq C \sqcap D}$ $b_{A \sqsubseteq C}$ $\mathbf{b}_{B \sqsubseteq C}$ $b_{A \sqsubseteq \perp}$ $b_{A \sqsubseteq B \sqcap C}$	r_2	$\neg \mathbf{b}_{B \sqsubseteq C}$
$b_{A \sqsubseteq C \sqcap D}$ $\mathbf{b}_{A \sqsubseteq D}$ $b_{A \sqsubseteq \perp}$ $\mathbf{b}_{B \sqcap C \sqsubseteq \perp}$	r_3	$\neg \mathbf{b}_{A \sqsubseteq D} \vee \neg \mathbf{b}_{B \sqcap C \sqsubseteq \perp}$

The first column lists all variables corresponding to the broken axioms. For the set M of the remaining axioms from \mathcal{O} , our encoding guarantees that $M \not\models \alpha$. In our example, the sets M are already maximal, so their complements (Line 5) correspond to repairs shown in the second column. Each of these repairs produces a new clause shown in the last column, which is added as a conjunction to F . The algorithm, therefore, produces overall $7 + 3 = 10$ clauses.

The main advantage of Algorithm 2 over Algorithm 1 is that generation of subsets of $M \subseteq \mathcal{O}$ with $M \models \alpha$ is completely avoided. The initial formula F obtained by $\text{InitRepairs}(\alpha, \mathsf{l})$, which has a linear size in the number of inferences in l , has the same effect as (potentially exponentially-many in the size of \mathcal{O}) blocking clauses for justifications added to F in Algorithm 2. A disadvantage of Algorithm 2 is that the new SAT encoding now involves more propositional variables (due to new axioms appearing in inferences), which could increase the search space of the SAT solver. Also, if the set of inferences l is considerably larger than the size of \mathcal{O} , the new translation might not pay off, particularly, if the number of justifications is relatively small.

5. Computing Just Justifications

We now extend the ideas used in the encoding for repairs in Section 4, to compute all justifications without computing repairs. Our goal is to construct a formula F whose models correspond to only subsets $M \subseteq \mathcal{O}$ such that $M \models \alpha$, or, equivalently, $M \vdash_1 \alpha$. Recall that α is derivable by inferences in \mathbb{I} from \mathcal{O} if (1) $\alpha \in \mathcal{O}$, or (2) there exists an inference $\langle \beta_1, \dots, \beta_n \models \alpha \rangle \in \mathbb{I}$ such that all premises β_1, \dots, β_n of this inference are derivable. To handle Condition (1) as a special case of (2), we extend the set \mathbb{I} by adding, for every axiom $\beta \in \mathcal{O}$, an “asserted axiom” inference $\langle \vdash \beta \rangle$, which means that any axiom in \mathcal{O} can be derived from no premises.

To express Condition (2) using a propositional formula, to each axiom β appearing in \mathbb{I} , we assign a distinguished propositional variable d_β expressing that β appears in the derivation of α . In addition, to every inference $\text{inf} \in \mathbb{I}$ we, likewise, assign a distinguished variable d_{inf} expressing that this inference has been applied in the derivation of α . Then Condition (2) is expressed using two types of clauses:

1. $\neg d_\beta \vee d_{\text{inf}_1} \vee \dots \vee d_{\text{inf}_m}$, where $\text{inf}_1, \dots, \text{inf}_m$ are all inferences in \mathbb{I} with conclusion β ,
2. $\neg d_{\text{inf}} \vee d_\beta$, where β is a premise of inf .

Let F be the conjunction of all clauses above for all axioms β and inferences inf appearing in \mathbb{I} plus the unit clause d_α expressing that α must be derived. Similarly to the encoding for repairs in Section 4, we define a subset $M \subseteq \mathcal{O}$ using a model \mathcal{I} of F by $M = M(\mathcal{I}) = \{\beta \in \mathcal{O} \mid d_\beta^{\mathcal{I}} = 1\}$. The main question: does our encoding guarantee that $M \models \alpha$?

Example 6. Let us apply the described translation for the entailment $\mathcal{O} \models \alpha$ from Example 1 and the set of inferences \mathbb{I} from Example 2. First, we extend \mathbb{I} with inferences deriving axioms in \mathcal{O} : $\text{inf}_7: \langle \vdash A \sqsubseteq B \rangle$, $\text{inf}_8: \langle \vdash B \sqsubseteq C \rangle$, $\text{inf}_9: \langle \vdash A \sqsubseteq D \rangle$, $\text{inf}_{10}: \langle \vdash B \sqcap C \sqsubseteq \perp \rangle$.

Then our encoding produces a formula F , which is the conjunction of the following 20 clauses:

- | | | |
|--|---|--|
| 1. $d_{A \sqsubseteq C \sqcap D}$ | 8. $\neg \mathbf{d}_{B \sqsubseteq C} \vee d_{\text{inf}_8}$ | 15. $\neg d_{\text{inf}_3} \vee \mathbf{d}_{A \sqsubseteq B}$ |
| 2. $\neg d_{A \sqsubseteq C} \vee d_{\text{inf}_1}$ | 9. $\neg \mathbf{d}_{A \sqsubseteq D} \vee d_{\text{inf}_9}$ | 16. $\neg d_{\text{inf}_3} \vee d_{A \sqsubseteq C}$ |
| 3. $\neg d_{A \sqsubseteq C \sqcap D} \vee d_{\text{inf}_2} \vee d_{\text{inf}_6}$ | 10. $\neg \mathbf{d}_{B \sqcap C \sqsubseteq \perp} \vee d_{\text{inf}_{10}}$ | 17. $\neg d_{\text{inf}_4} \vee d_{A \sqsubseteq B \sqcap C}$ |
| 4. $\neg d_{A \sqsubseteq B \sqcap C} \vee d_{\text{inf}_3}$ | 11. $\neg d_{\text{inf}_1} \vee \mathbf{d}_{A \sqsubseteq B}$ | 18. $\neg d_{\text{inf}_4} \vee \mathbf{d}_{B \sqcap C \sqsubseteq \perp}$ |
| 5. $\neg d_{A \sqsubseteq \perp} \vee d_{\text{inf}_4}$ | 12. $\neg d_{\text{inf}_1} \vee \mathbf{d}_{B \sqsubseteq C}$ | 19. $\neg d_{\text{inf}_6} \vee d_{A \sqsubseteq \perp}$ |
| 6. $\neg d_{\perp \sqsubseteq C \sqcap D} \vee d_{\text{inf}_5}$ | 13. $\neg d_{\text{inf}_2} \vee d_{A \sqsubseteq C}$ | 20. $\neg d_{\text{inf}_6} \vee d_{\perp \sqsubseteq C \sqcap D}$ |
| 7. $\neg \mathbf{d}_{A \sqsubseteq B} \vee d_{\text{inf}_7}$ | 14. $\neg d_{\text{inf}_2} \vee \mathbf{d}_{A \sqsubseteq D}$ | |

As before, bold variables correspond to axioms from \mathcal{O} . By checking satisfiability of F , we may obtain the models shown in the first column of the following table:

$d_\beta^{\mathcal{I}} = 1$	justification	new clause
$d_{A \sqsubseteq C \sqcap D} \ d_{\text{inf}_2} \ d_{A \sqsubseteq C} \ \mathbf{d}_{A \sqsubseteq D} \ d_{\text{inf}_1}$ $d_{\text{inf}_9} \ \mathbf{d}_{A \sqsubseteq B} \ \mathbf{d}_{B \sqsubseteq C} \ d_{\text{inf}_7} \ d_{\text{inf}_8}$	j_1	$\neg \mathbf{d}_{A \sqsubseteq D} \vee \neg \mathbf{d}_{A \sqsubseteq B} \vee \neg \mathbf{d}_{B \sqsubseteq C}$
$d_{A \sqsubseteq C \sqcap D} \ d_{\text{inf}_6} \ d_{A \sqsubseteq \perp} \ d_{\perp \sqsubseteq C \sqcap D} \ d_{\text{inf}_4}$ $d_{\text{inf}_5} \ d_{A \sqsubseteq B \sqcap C} \ \mathbf{d}_{B \sqcap C \sqsubseteq \perp} \ d_{\text{inf}_3} \ d_{\text{inf}_{10}}$ $\mathbf{d}_{A \sqsubseteq B} \ d_{A \sqsubseteq C} \ d_{\text{inf}_1} \ \mathbf{d}_{B \sqsubseteq C}$	j_2	$\neg \mathbf{d}_{A \sqsubseteq B} \vee \neg \mathbf{d}_{B \sqsubseteq C} \vee \neg \mathbf{d}_{B \sqcap C \sqsubseteq \perp}$

The sets $M = M(\mathcal{I})$ extracted from these models consist of axioms corresponding to the bold variables and, as can be seen, correspond to justifications j_1 and j_2 from Example 1. If we now extend F by adding the corresponding blocking clauses for these justifications shown in the third column, F becomes unsatisfiable. Hence, no further sets M and justifications can be produced.

Example 7. Let us see how the new translation can avoid computing exponentially-many repairs for the entailment $\mathcal{O} \models A \sqsubseteq C$ from Example 4. Assume that we are given a set of inferences $\mathsf{l} = \{\text{inf}_i = \langle A \sqsubseteq B_i, B_i \sqsubseteq C \vdash A \sqsubseteq C \rangle \mid 1 \leq i \leq n\}$, which is, clearly, complete for this entailment. First, l is extended with $\text{inf}'_i = \langle \vdash A \sqsubseteq B_i \rangle$ and $\text{inf}''_i = \langle \vdash B_i \sqsubseteq C \rangle$ ($1 \leq i \leq n$). The translation creates Clauses 1 and 2 plus Clauses 3–6 for each i with $1 \leq i \leq n$:

1. $d_{A \sqsubseteq C}$
2. $\neg d_{A \sqsubseteq C} \vee \bigvee \{d_{\text{inf}_i} \mid 1 \leq i \leq n\}$
3. $\neg d_{\text{inf}_i} \vee \mathbf{d}_{A \sqsubseteq B_i}$
4. $\neg d_{\text{inf}_i} \vee \mathbf{d}_{B_i \sqsubseteq C}$
5. $\neg \mathbf{d}_{A \sqsubseteq B_i} \vee d_{\text{inf}'_i}$
6. $\neg \mathbf{d}_{B_i \sqsubseteq C} \vee d_{\text{inf}''_i}$

After each call, due to Clauses 1–4, the SAT solver returns a model with $d_{\text{inf}_i}^{\mathcal{I}} = d_{A \sqsubseteq B_i}^{\mathcal{I}} = d_{B_i \sqsubseteq C}^{\mathcal{I}} = 1$ for some i ($1 \leq i \leq n$) from which the justification $j_i = \{A \sqsubseteq B_i, B_i \sqsubseteq C\}$ can be extracted. The added blocking clause $\neg d_{A \sqsubseteq B_i} \vee \neg d_{B_i \sqsubseteq C}$ ensures that $\text{inf}_i^{\mathcal{I}} = 0$ in the subsequent models \mathcal{I} (due to Clauses 3 and 4). Hence, the procedure terminates after adding exactly n additional clauses.

Examples 6 and 7 may suggest that $M \models \alpha$ always holds for subsets M extracted from models of F . However, this is true only if the set of inferences l is *acyclic*. An *inference cycle* (of length n) is a sequence of inferences $[\text{inf}_1; \text{inf}_2; \dots; \text{inf}_n]$ such that the conclusion of inf_i is a premise of inf_{i+1} ($1 \leq i < n$) and the conclusion of inf_n is a premise of inf_1 . The cycle is *elementary* if all of its inferences have different conclusions. Clearly, each (non-elementary) cycle $[\text{inf}_1; \text{inf}_2; \dots; \text{inf}_n]$ contains an elementary *sub-cycle* $[\text{inf}_k; \dots; \text{inf}_m]$ with $1 \leq k \leq m \leq n$. A set of inferences l is *cyclic* if there is a cycle of inferences from l . Otherwise l is *acyclic*.

Example 8. Let us extend the set of inferences l from Example 6 with two additional inferences: $\text{inf}_{11}: \langle A \sqsubseteq B \sqcap C \vdash A \sqsubseteq B \rangle$, $\text{inf}_{12}: \langle A \sqsubseteq B \sqcap C \vdash A \sqsubseteq C \rangle$.

The resulting set of inferences has 2 elementary cycles: $c_1 = [\text{inf}_3; \text{inf}_{11}]$ and $c_2 = [\text{inf}_3; \text{inf}_{12}]$. Updating the encoding of the formula F from Example 6 to the two additional inferences results in two modified clauses 2, 7, and two additional clauses 20, 21 shown below:

2. $\neg d_{A \sqsubseteq C} \vee d_{\text{inf}_1} \vee d_{\text{inf}_{12}}$
7. $\neg \mathbf{d}_{A \sqsubseteq B} \vee d_{\text{inf}_7} \vee d_{\text{inf}_{11}}$
20. $\neg d_{\text{inf}_{11}} \vee \mathbf{d}_{A \sqsubseteq B \sqcap C}$
21. $\neg d_{\text{inf}_{12}} \vee \mathbf{d}_{A \sqsubseteq B \sqcap C}$

The models of F obtained in Example 6 are also models of the new clause set, however, there is also a new model shown below:

$d_{\beta}^{\mathcal{I}} = 1$	cycle	new clause
$d_{A \sqsubseteq C \sqcap D} \quad d_{\text{inf}_2} \quad d_{A \sqsubseteq C} \quad \mathbf{d}_{A \sqsubseteq D} \quad d_{\text{inf}_{12}}$	c_2	$\neg \text{inf}_3 \vee \neg \text{inf}_{12}$
$d_{\text{inf}_9} \quad d_{A \sqsubseteq B \sqcap C} \quad d_{\text{inf}_3} \quad \mathbf{d}_{A \sqsubseteq B} \quad d_{\text{inf}_7}$		

Notice that $M = M(\mathcal{I}) = \{A \sqsubseteq D, A \sqsubseteq B\} \not\models \alpha = A \sqsubseteq C \sqcap D$, so why did this happen in our translation? Note that the new model is similar to the first model in Example 6. The main difference is that to satisfy the (updated) clause 2, we now chose to make $d_{\text{inf}_{12}}$ true instead of d_{inf_1} .

Algorithm 3: Computing all justifications using a SAT solver

SAT-Justification($\mathcal{O}, \alpha, \mathsf{l}$):
input : ontology \mathcal{O} , α an axiom, l a set of sound inferences complete for $\mathcal{O} \models \alpha$
output : the set J of all justifications for $\mathcal{O} \models \alpha$

```
1  $J \leftarrow \emptyset$ ;  
2  $\mathsf{l} \leftarrow \mathsf{l} \cup \{\langle \vdash \beta \rangle \mid \beta \in \mathcal{O}\}$ ;  
3  $F \leftarrow \text{InitJustifications}(\alpha, \mathsf{l})$ ;  
4  $F \leftarrow F \wedge \text{InitCycles}(\mathsf{l})$ ;  
5 while  $\{\mathcal{I} \mid F^{\mathcal{I}} = 1\} \neq \emptyset$  do  
6    $\mathcal{I} \leftarrow \mathcal{I} : F^{\mathcal{I}} = 1$ ;  
7    $M \leftarrow \{\beta \in \mathcal{O} \mid d_{\beta}^{\mathcal{I}} = 1\}$ ;  
8   if  $M \models \alpha$  then  
9      $j \leftarrow \text{Minimize}(M \models \alpha)$ ;  
10     $J \leftarrow J \cup \{j\}$ ;  
11     $F \leftarrow F \wedge \bigvee \{\neg d_{\beta} \mid \beta \in j\}$ ;  
12  else  
13     $\mathsf{l}_o \leftarrow \{\text{inf} \in \mathsf{l} \mid \text{inf}^{\mathcal{I}} = 1\}$ ;  
14     $c \leftarrow \text{FindCycle}(\alpha, \mathsf{l}_o)$ ;  
15     $F \leftarrow F \wedge \bigvee \{\neg d_{\text{inf}} \mid \text{inf} \in c\}$ ;  
16 return  $J$ ;
```

```
17  $F \leftarrow d_{\alpha}$ ;  
18 foreach  $\beta$  occurring in  $\mathsf{l}$  do  
19    $F \leftarrow F \wedge$   
     $(\neg d_{\beta} \vee \bigvee \{\text{inf} = \langle \dots \vdash \beta \rangle \in \mathsf{l}\})$ ;  
20 foreach  $\text{inf} = \langle \beta_1, \dots, \beta_n \vdash \beta \rangle \in \mathsf{l}$  do  
21    $F \leftarrow F \wedge \bigwedge_{1 \leq i \leq n} (\neg d_{\text{inf}} \vee d_{\beta_i})$ ;  
22 return  $F$ ;
```

FindCycle(α, l_o):
23 Visited $\leftarrow \emptyset$, $L \leftarrow []$, $\beta \leftarrow \alpha$;
24 **while** $\beta \notin \text{Visited}$ **do**
25 Visited $\leftarrow \text{Visited} \cup \{\beta\}$;
26 $\text{inf} \leftarrow \langle \beta_1, \dots, \beta_n \vdash \beta \rangle \in \mathsf{l}_o$;
27 $L \leftarrow [\text{inf}] + L$;
28 $\beta \leftarrow \beta_i : \emptyset \not\vdash_{\mathsf{l}_o} \beta_i (1 \leq i \leq n)$;
29 **return** $[\dots; \text{inf}_i] \subseteq L : \text{inf}_i = \langle \dots \vdash \beta \rangle$;

This, intuitively, means that we chose to derive $A \sqsubseteq C$ by inference inf_{12} instead of inf_1 . To apply inf_{12} , we need to derive $A \sqsubseteq B \sqcap C$ first, so we make $d_{A \sqsubseteq B \sqcap C}$ true. This axiom, in turn, can be derived using inf_3 from $A \sqsubseteq B$ and $A \sqsubseteq C$, so we make d_{inf_3} , $d_{A \sqsubseteq B}$ and $d_{A \sqsubseteq C}$ true. However, we now came back to $A \sqsubseteq C$. This happened because of the inference cycle $c_2 = [\text{inf}_3; \text{inf}_{12}]$.

We can prevent generating models with cycle c_2 by adding a blocking clause similar to blocking clauses for justifications shown in the last column. After adding this clause to F the resulting formula becomes unsatisfiable, at which point we know that all justifications have been computed.

Algorithm 3 formalizes the overall procedure for computing justifications. We start with the empty set of justifications (Line 1), and extend the set of inferences l for axioms in \mathcal{O} (Line 2). Then we initialize our formula F (Line 3) by adding the clauses for α (Line 17), for each axiom in l (Line 19), and each inference in l (Line 21). For now, ignore Line 4 assuming that $\text{InitCycles}(\mathsf{l})$ returns \top . Then, as usual, so long F is satisfiable (Line 5), we take the model \mathcal{I} returned by the SAT solver (Line 6), determine the candidate subset of axioms $M = M(\mathcal{I}) \subseteq \mathcal{O}$ (Line 7), and check the entailment $M \models \alpha$ (Line 8). If the entailment holds, we extract a new justification from M as usual (Lines 8–11). If not, this can happen only due to a cyclic subset of inferences l_o chosen by the model (Line 13), as we prove next. Note that $M \not\models \alpha$ implies $\emptyset \not\vdash_{\mathsf{l}_o} \alpha$.

To find a cycle in l_o (Line 14), we traverse these inferences backwards starting from the conclusion α . To detect the cycle, we add conclusions to the set Visited, while collecting the expanded inferences in the list L , both initially empty (Line 23). Our invariant is that $d_{\beta}^{\mathcal{I}} = 1$ but

$\emptyset \not\vdash_{I_0} \beta$ for every $\beta \in \text{Visited}$, which clearly holds for the initial $\beta = \alpha$. Having such β , we first check that it was not already visited (Line 24) and add it to Visited (Line 25). Then we take any inference $\text{inf} \in I_0$ with conclusion β (Line 26), which must exist due to the added clause for β (Line 19) since $F^{\mathcal{I}} = 1$ and $d_{\beta}^{\mathcal{I}} = 1$. We prepend this inference to the list L (Line 27). Finally, we choose the premise β_i of inf such that $\emptyset \not\vdash_{I_0} \beta_i$. Note that such a premise exists since otherwise $\emptyset \vdash_{I_0} \beta$, which would contradict our invariant. Note also that $d_{\beta_i}^{\mathcal{I}} = 1$ for all premises of inf due to the initial translation for inf (Line 21). We reassign β to the found premise β_i (Line 28) and continue the while loop. Eventually we obtain $\beta \in \text{Visited}$ since the number of axioms is bounded, which means that a premise β of the first inference in L is a conclusion of some inference inf_i in L . We traverse the list L to find this inference inf_i , and return the prefix of L ending with this inference (Line 29), which will be the elementary inference cycle we were looking for. After finding this cycle, we add the corresponding blocking clause to F (Line 11), which makes sure that this cycle will not be a part of subsequent models of F .

How many cycles could be encountered during a run of Algorithm 3? Since $\text{FindCycle}(\alpha, I_0)$ produces only elementary cycles, the length of cycles is bounded by the number of inferences in I . Further, the set of inferences in each returned cycle is unique due to the added blocking clause in Line 11. Hence the number of cycles and such blocking clauses is at most *exponential* in the size of I . It may seem that Algorithm 3 suffers from the same problem as Algorithm 1: instead of enumerating possibly exponentially-many repairs we may need to enumerate possibly exponentially-many cycles in order to compute all justifications. As we show next, it is possible to extend the initial formula F of the encoding to prevent cyclic models completely.

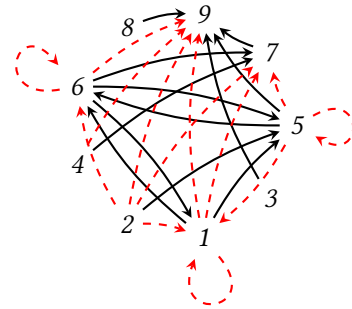
6. Blocking of Cycles using a Transitive Closure Encoding

Let I be a set of inferences. Consider the binary relation on axioms appearing in I defined by the set of pairs $E = E(I) = \{\langle \beta_i, \beta \rangle \mid \langle \beta_1, \dots, \beta_n \vdash \beta \rangle \in I, 1 \leq i \leq n\}$. It is easy to see that I is cyclic iff the transitive closure E^+ of E contains a *reflexive pair* $\langle \beta, \beta \rangle$ for some axiom β .

Example 9. Let I be the set of (cyclic) inferences from Example 8. For conciseness, we number all axioms appearing in I and use these numbers instead of the original axioms:

$$\begin{array}{lll} A \sqsubseteq B \rightsquigarrow 1 & B \sqcap C \rightsquigarrow 4 & A \sqsubseteq \perp \rightsquigarrow 7 \\ B \sqsubseteq C \rightsquigarrow 2 & A \sqsubseteq C \rightsquigarrow 5 & \perp \sqsubseteq C \sqcap D \rightsquigarrow 8 \\ A \sqsubseteq D \rightsquigarrow 3 & A \sqsubseteq B \sqcap C \rightsquigarrow 6 & A \sqsubseteq C \sqcap D \rightsquigarrow 9 \end{array}$$

Then the set of inferences I defines the set $E = E(I)$ of pairs of numbers corresponding to these axioms shown in the right picture by solid lines. As can be seen, nodes 1 and 6 as well as nodes 5 and 6 form cycles in this graph. The transitive closure E^+ of E extends this relation with the additional pairs shown by the dashed red lines. As can be seen, the extended graph contains 3 reflexive pairs: $\langle 1, 1 \rangle$, $\langle 5, 5 \rangle$, and $\langle 6, 6 \rangle$.



Let $E = E(I)$ be the set of pairs induced by inferences from I . For each pair $\langle \alpha, \beta \rangle \in E^+$ we introduce a distinguished propositional variable $d_{\langle \alpha, \beta \rangle}$. Consider the following clauses:

1. $\neg d_{\text{inf}} \vee d_{\langle \beta_i, \beta \rangle}$ for every $\text{inf} = \langle \beta_1, \dots, \beta_n \vdash \beta \rangle \in \mathcal{I}$ and $1 \leq i \leq n$,
2. $\neg d_{\langle \alpha, \beta \rangle} \vee \neg d_{\langle \beta, \gamma \rangle} \vee d_{\langle \alpha, \gamma \rangle}$ for every $\langle \alpha, \beta \rangle \in E^+$ and $\langle \beta, \gamma \rangle \in E$,
3. $\neg d_{\langle \alpha, \alpha \rangle}$ for every $\langle \alpha, \alpha \rangle \in E^+$.

Intuitively, clauses of Form 1 express that if an inference $\langle \beta_1, \dots, \beta_n \vdash \beta \rangle \in \mathcal{I}$ is selected by the model, then all pairs $\langle \beta_i, \beta \rangle \in E$ ($1 \leq i \leq n$) must be selected by this model as well. Clauses of Form 2 express that the pairs selected by the model are transitively closed. Finally, clauses of Form 3 express that no reflexive pair from E^+ can be selected by the model.

Suppose that the function `InitCycles(I)` produces the conjunction of all clauses of Forms 1-3 as described above. By using this function in Line 4 of Algorithm 3, we make sure that for every model \mathcal{I} of F , the set of inferences $\mathcal{I} = \{\text{inf} \in \mathcal{I} \mid d_{\text{inf}}^{\mathcal{I}} = 1\}$ is acyclic. Hence the else-part of the algorithm (Lines 12–15) never applies, and each model found by the SAT solver corresponds to a new justification. Clearly, the number of clauses of Form 1 is linear in the size of \mathcal{I} . Since the number of axioms α appearing in \mathcal{I} and the number of pairs $\langle \beta, \gamma \rangle \in E$ are linear in the size of \mathcal{I} , the number of clauses of Form 2 is *quadratic* in the size of \mathcal{I} and the number of clauses of Form 3 is linear. Thus, the maximal number of clauses generated by Algorithm 3 can be reduced from exponential to quadratic.

7. Outlook

In this paper, we have described new SAT-based procedures for computing repairs and justifications from a set of inferences. Unlike the previous methods, we use inferences not only for entailment checks, but also to avoid generating unnecessary models. While for computing repairs our encoding requires a linear number of additional clauses, computing justifications may require a quadratic number of clauses in order to block cyclic models.

There appears to be a strong connection between our encodings and reductions of Answer Set Programming (ASP) to SAT [29, 30, 31, 32, 33, 34]. In particular, our encoding for computing justifications is closely related to Clark’s Completion [35], which bridges the ASP semantics and the classical (model-theoretic) semantics. Just like in our case, Clark’s Completion works only for acyclic logic programs called *tight* [36]. Cyclic programs are dealt with using *one shot* quadratic [29, 30] or $O(n \cdot \log_2 n)$ encodings [33, 34], or using (possibly exponentially-many) *looping formulas* that block cycles only when they are discovered [31, 32].

For some ontologies, such as OpenGALEN [37], for which the generated proofs can be very large and cyclic [16], a quadratic number of clauses can still be prohibitively large. To reduce the number of clauses even further, it is possible to adapt the $O(n \cdot \log_2 n)$ encoding for ASP mentioned above, and use further optimizations, such as *lazy encoding*, which delays creation of clauses until cycles are actually found.

By combining encoding for repairs and justifications, it is also possible to obtain a procedure for computing the unions of justifications (= the unions of repairs), which were recently of some interest [38, 39, 40]. This procedure can compute the unions using at most n calls of the SAT solver, where n is the size of the resulting union. This is optimal since the membership problem for unions of justifications is already NP-Complete (see Theorem 7 in [41]). Our preliminary empirical experiments on some selected difficult examples have been very promising. Unfortunately, we have no space left in this paper to discuss these results in detail.

References

- [1] F. Baader, C. Lutz, B. Suntisrivaraporn, CEL—a polynomial-time reasoner for life science ontologies, in: Proc. 3rd Int. Joint Conf. on Automated Reasoning (IJCAR’06), volume 4130 of *LNCS*, Springer, 2006, pp. 287–291.
- [2] Y. Kazakov, M. Krötzsch, F. Simančík, The incredible ELK: From polynomial procedures to efficient reasoning with \mathcal{EL} ontologies, *J. of Automated Reasoning* 53 (2014) 1–61.
- [3] D. Tsarkov, I. Horrocks, FaCT++ description logic reasoner: System description, in: Proc. 3rd Int. Joint Conf. on Automated Reasoning (IJCAR’06), volume 4130 of *LNCS*, Springer, 2006, pp. 292–297.
- [4] B. Motik, R. Shearer, I. Horrocks, Hypertableau reasoning for description logics, *J. of Artificial Intelligence Research* 36 (2009) 165–228.
- [5] A. Steigmiller, T. Liebig, B. Glimm, Konclude: System description, *J. Web Semant.* 27-28 (2014) 78–85.
- [6] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, Y. Katz, Pellet: A practical OWL-DL reasoner, *J. of Web Semantics* 5 (2007) 51–53.
- [7] F. Baader, R. Peñaloza, B. Suntisrivaraporn, Pinpointing in the description logic \mathcal{EL}^+ , in: J. Hertzberg, M. Beetz, R. Englert (Eds.), Proc. 30th Annual German Conf. on Artificial Intelligence (KI’07), volume 4667 of *LNCS*, Springer, 2007, pp. 52–67. URL: https://doi.org/10.1007/978-3-540-74565-5_7. doi:10.1007/978-3-540-74565-5_7.
- [8] A. Kalyanpur, B. Parsia, M. Horridge, E. Sirin, Finding all justifications of OWL DL entailments, in: ISWC/ASWC, volume 4825 of *Lecture Notes in Computer Science*, Springer, 2007, pp. 267–280.
- [9] R. Sebastiani, M. Vescovi, Axiom pinpointing in lightweight description logics via Horn-SAT encoding and conflict analysis, in: R. A. Schmidt (Ed.), Proc. 22st Conf. on Automated Deduction (CADE’09), volume 5663 of *LNCS*, Springer, 2009, pp. 84–99. URL: http://dx.doi.org/10.1007/978-3-642-02959-2_6. doi:10.1007/978-3-642-02959-2_6.
- [10] F. Baader, R. Peñaloza, Automata-based axiom pinpointing, *J. Autom. Reason.* 45 (2010) 91–129.
- [11] F. Baader, R. Peñaloza, Axiom pinpointing in general tableaux, *J. Log. Comput.* 20 (2010) 5–34.
- [12] M. Horridge, Justification based explanation in ontologies, Ph.D. thesis, University of Manchester, UK, 2011. URL: <http://www.manchester.ac.uk/escholar/uk-ac-man-scw:131699>.
- [13] M. F. Arif, C. Mencía, J. Marques-Silva, Efficient axiom pinpointing with EL2MCS, in: S. Hölldobler, M. Krötzsch, R. Peñaloza, S. Rudolph (Eds.), Proc. 38th Annual German Conf. on Artificial Intelligence (KI’15), volume 9324 of *LNCS*, Springer, 2015, pp. 225–233. URL: http://dx.doi.org/10.1007/978-3-319-24489-1_17. doi:10.1007/978-3-319-24489-1_17.
- [14] M. F. Arif, C. Mencía, J. Marques-Silva, Efficient MUS enumeration of horn formulae with applications to axiom pinpointing, CoRR abs/1505.04365 (2015). URL: <http://arxiv.org/abs/1505.04365>.
- [15] N. Manthey, R. Peñaloza, S. Rudolph, Efficient axiom pinpointing in \mathcal{EL} using SAT technology, in: M. Lenzerini, R. Peñaloza (Eds.), Proc. 29th Int. Workshop on Description

- Logics (DL'16), volume 1577 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2016. URL: http://ceur-ws.org/Vol-1577/paper_33.pdf.
- [16] Y. Kazakov, P. Skocovský, Enumerating justifications using resolution, in: *IJCAR*, volume 10900 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 609–626.
 - [17] A. Ozaki, R. Peñaloza, Consequence-based axiom pinpointing, in: *SUM*, volume 11142 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 181–195.
 - [18] N. Manthey, R. Peñaloza, S. Rudolph, SATPin: Axiom pinpointing for lightweight description logics through incremental SAT, *Künstliche Intell.* 34 (2020) 389–394.
 - [19] F. Baader, S. Brandt, C. Lutz, Pushing the \mathcal{EL} envelope, in: *Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05)*, 2005, pp. 364–369.
 - [20] Y. Kazakov, Consequence-driven reasoning for Horn \mathcal{SHIQ} ontologies, in: *Proc. 21st Int. Joint Conf. on Artificial Intelligence (IJCAI'09)*, *IJCAI*, 2009, pp. 2040–2045.
 - [21] F. Simancik, Y. Kazakov, I. Horrocks, Consequence-based reasoning beyond horn ontologies, in: *Proc. 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11)*, 2011, pp. 1093–1098.
 - [22] A. Bate, B. Motik, B. C. Grau, D. T. Cucala, F. Simancik, I. Horrocks, Consequence-based reasoning for description logics with disjunctions and number restrictions, *J. Artif. Intell. Res.* 63 (2018) 625–690.
 - [23] D. T. Cucala, B. C. Grau, I. Horrocks, Sequoia: A consequence based reasoner for SROIQ, in: *Description Logics*, volume 2373 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2019.
 - [24] F. Baader, I. Horrocks, C. Lutz, U. Sattler, *An Introduction to Description Logic*, Cambridge University Press, 2017.
 - [25] M. H. Liffiton, A. Previti, A. Malik, J. Marques-Silva, Fast, flexible MUS enumeration, *Constraints An Int. J.* 21 (2016) 223–250.
 - [26] M. Vescovi, *Exploiting SAT and SMT Techniques for Automated Reasoning and Ontology Manipulation in Description Logics*, Ph.D. thesis, University of Trento, Italy, 2011. URL: <http://eprints-phd.biblio.unitn.it/477/>.
 - [27] M. F. Arif, C. Mencía, A. Ignatiev, N. Manthey, R. Peñaloza, J. Marques-Silva, BEACON: an efficient sat-based tool for debugging \mathcal{EL}^+ ontologies, in: N. Creignou, D. L. Berre (Eds.), *Proc. 19th Int. Conf. on Theory and Applications of Satisfiability Testing (SAT'16)*, volume 9710 of *LNCS*, Springer, 2016, pp. 521–530. URL: http://dx.doi.org/10.1007/978-3-319-40970-2_32. doi:10.1007/978-3-319-40970-2_32.
 - [28] B. Glimm, Y. Kazakov, Classical algorithms for reasoning and explanation in description logics, in: *Reasoning Web*, volume 11810 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 1–64.
 - [29] R. Ben-Eliyahu, R. Dechter, Propositional semantics for disjunctive logic programs, *Ann. Math. Artif. Intell.* 12 (1994) 53–87.
 - [30] F. Lin, J. Zhao, On tight logic programs and yet another translation from normal logic programs to propositional logic, in: *IJCAI*, Morgan Kaufmann, 2003, pp. 853–858.
 - [31] F. Lin, Y. Zhao, ASSAT: computing answer sets of a logic program by SAT solvers, *Artif. Intell.* 157 (2004) 115–137.
 - [32] E. Giunchiglia, Y. Lierler, M. Maratea, Answer set programming based on propositional satisfiability, *J. Autom. Reason.* 36 (2006) 345–377.
 - [33] T. Janhunen, Representing normal programs with clauses, in: *ECAI*, IOS Press, 2004, pp. 358–362.

- [34] T. Janhunen, Some (in)translatability results for normal logic programs and propositional theories, *J. Appl. Non Class. Logics* 16 (2006) 35–86.
- [35] K. L. Clark, Negation as failure, in: H. Gallaire, J. Minker (Eds.), *Logic and Data Bases, Symposium on Logic and Data Bases, Centre d'études et de recherches de Toulouse, France, 1977, Advances in Data Base Theory*, Plenum Press, New York, 1977, pp. 293–322. URL: https://doi.org/10.1007/978-1-4684-3384-5_11. doi:10.1007/978-1-4684-3384-5_11.
- [36] F. Fages, Consistency of Clark's completion and existence of stable models, *Methods Log. Comput. Sci.* 1 (1994) 51–60.
- [37] A. L. Rector, J. Rogers, P. E. Zanstra, E. J. van der Haring, OpenGALEN: Open source medical terminology and tools, in: *AMIA, AMIA, 2003*.
- [38] M. Janota, J. Marques-Silva, cmmus: A tool for circumscription-based MUS membership testing, in: *LPNMR, volume 6645 of Lecture Notes in Computer Science*, Springer, 2011, pp. 266–271.
- [39] C. Mencía, O. Kullmann, A. Ignatiev, J. Marques-Silva, On computing the union of MUSes, in: *SAT, volume 11628 of Lecture Notes in Computer Science*, Springer, 2019, pp. 211–221.
- [40] J. Chen, Y. Ma, R. Peñaloza, H. Yang, Union and intersection of all justifications, in: *ESWC, volume 13261 of Lecture Notes in Computer Science*, Springer, 2022, pp. 56–73.
- [41] R. Peñaloza, B. Sertkaya, Understanding the complexity of axiom pinpointing in lightweight description logics, *Artif. Intell.* 250 (2017) 80–104.