Axiom Pinpointing Using an Assumption-Based Truth Maintenance System

Hai Nguyen, Natasha Alechina, and Brian Logan

University of Nottingham

1 Introduction

The problem of *axiom pinpointing* [1, 22], that is, finding the minimal set of axioms responsible for an unwanted consequence, is an important problem in ontology debugging. One approach to identifying the axioms responsible for an unwanted consequence is to trace dependencies between inferences leading to the consequence. Several authors have proposed *truth maintenance systems* as a means of keeping track of dependencies or inferences in ontologies, e.g., [21, 5, 7]. In this paper we show that truth maintenance systems can also be used for axiom pinpointing. More specifically, we present a system which returns all minimal sets of axioms responsible for the derivation of inconsistency in an unfoldable $\mathcal{ALC}$ ontology. Following Sirin *et al* [24], we refer to these sets of axioms as *explanations*.

Our approach involves using a modified Assumption-Based Truth Maintenance System (ATMS) [11] to trace inferential dependencies between formulas and compute the minimal sets of ontology axioms responsible for a contradiction. The main technical contribution of the paper is extending the ATMS to deal with disjunctions. We generalise the notion of an ATMS environment (a set of axioms from which a formula is derivable) to include the non-deterministic choices required for the derivation of the formula. We show that this extended ATMS (which we call the D-ATMS), combined with a tableau reasoner, produces correct, complete and minimal explanations for a contradiction in an unfoldable $\mathcal{ALC}$ ontology. We have developed a prototype implementation of our approach which we call AOD. Preliminary results of experiments comparing AOD, MUPSter and the Pellet explanation service are encouraging, and suggest that AOD can outperform MUPSter and Pellet on both synthetic and real-world ontologies.

2 The Reasoner

Our ontology debugging framework, AOD, consists of two components: a tableau reasoner, and the D-ATMS (described in Section 3).

The reasoner takes a TBox as an input. To check for incoherence, we check whether a contradiction is derivable from the TBox and a statement of non-emptiness of a concept, eg $A(a)$. We refer to all TBox and ABox elements as formulas and reserve the term 'axiom' for the input formulas.

The reasoner is a tableau reasoner for $\mathcal{ALC}$ with unfoldable TBoxes [17, 2], and uses essentially the same rules as in [22, 16]:

$\sqsubseteq$**-rule** from $A(a)$ and $A \sqsubseteq C$ derive $C(a)$
$\sqcap$**-rule** from $(C_1 \sqcap \ldots \sqcap C_n)(a)$ derive $C_1(a), \ldots, C_n(a)$
$\exists$**-rule** from $(\exists s.C)(a)$ derive $s(a,b), C(b)$ where b is a new individual and $(\exists s.C)(a)$ has not been used before to generate another new individual
$\forall$**-rule** from $(\forall s.C)(a)$ and $s(a,b)$ derive $C(b)$
$\bot$**-rule** from $A(a)$ and $\neg A(a)$ derive $\bot$
$\sqcup$**-rule** from $(C_1 \sqcup \ldots \sqcup C_n)(a)$, derive choices $C_1(a), \ldots, C_n(a)$

where A is an atomic concept, C and D are arbitrary concepts, a, b are constants, and s is a role.

The $\sqcup$-rule creates branches in the tableaux for each disjunct (choice) $C_1(a), \ldots, C_n(a)$. A tableau is a tree where nodes are sets of formulas, and children of a node are obtained by applying inference rules to formulas in the node, so that the child node(s) contains all the formulas from the parent node and the newly derived formula. For readability, we will sometimes show only the new formula in a child node, with the understanding that all the formulas higher up on the branch belong to the node as well. If a node contains several disjunctions, for example $B_1 \sqcup B_2 \sqcup B_3(a)$ and $C_1 \sqcup C_2(b)$ as in Figure 1, the order in which the disjunction rule is applied does not matter, but once this order is fixed, the choices for the second disjunction are repeated under each of the choices for the first disjunction:

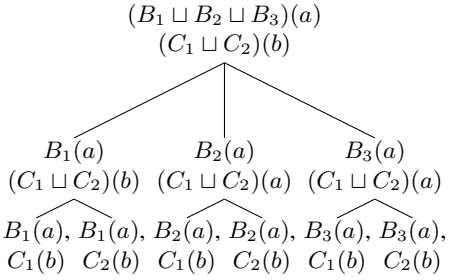

Fig. 1: Tableau with nested disjunctions

The reasoner derives consequences by applying inference rules to axioms and previously derived formulas. An *inference* $\phi_1, \ldots, \phi_n \stackrel{r}{\Longrightarrow} \phi$ indicates that the formula ϕ can be derived from the set of formulas $\phi_1, \ldots, \phi_n$ using the inference rule r. The reasoner does not stop after a contradiction is derived on a branch, but continues to apply inference rules until no new rule applications are possible. A rule application is new if the inference rule has not been used before with the same premises. The reasoner never repeats the same rule application.

3 The D-ATMS

The D-ATMS maintains dependencies between formulas inferred by the reasoner. To do so, the D-ATMS builds and maintains a *justification graph*. Each node in the graph corresponds to a formula or a justification. We denote the node corresponding to a formula

ϕ by n_ϕ. Axioms are represented by *axiom nodes*, and inconsistency is represented by a distinguished *false node*, $n_\bot$. A *justification* is a structure $j : n_{\phi_1}, \ldots, n_{\phi_k} \Rightarrow n_\phi$, where $n_{\phi_1}, \ldots, n_{\phi_k}$ are nodes corresponding to the antecedents of an inference rule application, n_ϕ is a node corresponding to the consequent,and j is the justification id, a unique, sequentially assigned integer that identifies the justification.[1] In the interests of readability, we will often refer to a formula node n_ϕ by the formula ϕ it represents.

When the reasoner applies an inference rule, it passes the resulting inference to the D-ATMS, causing the D-ATMS to update the justification graph. The reasoner keeps making inferences until no new inferences can be made. The D-ATMS is then invoked to compute all explanations for $\bot$. An *explanation* consists of all minimal sets of axioms from which $\bot$ can be derived, and, optionally, the sequence of inference rules necessary to derive $\bot$ from each set of axioms. The explanations returned by the D-ATMS are guaranteed to be correct (in the sense that $\bot$ is derivable from each of the returned sets of axioms) and minimal (in the sense that $\bot$ is not derivable from their proper subsets).

As an example, consider the following TBox inspired by the MadCow example from the OilEd tutorial:

ax1 $Sheep \sqsubseteq Animal$
ax2 $Cow \sqsubseteq Animal \sqcap \forall eats.\neg Animal$
ax3 $MadCow \sqsubseteq Cow \sqcap \exists eats.(Sheep \sqcup Cow)$

we also add the assumption $MadCow(a)$. The inferences made by the reasoner give rise to the following justifications (note that $Animal(b)$ has two justifications):

j_1 $MadCow(a), MadCow \sqsubseteq Cow \sqcap \exists eats.(Sheep \sqcup Cow) \Rightarrow Cow \sqcap \exists eats.(Sheep \sqcup Cow)(a)$
j_2 $Cow \sqcap \exists eats.(Sheep \sqcup Cow)(a) \Rightarrow Cow(a)$
j_3 $Cow \sqcap \exists eats.(Sheep \sqcup Cow)(a) \Rightarrow \exists eats.(Sheep \sqcup Cow)(a)$
j_4 $Cow(a), Cow \sqsubseteq Animal \sqcap \forall eats.\neg Animal, \Rightarrow Animal \sqcap \forall eats.\neg Animal(a)$
j_5 $\exists eats.(Sheep \sqcup Cow)(a) \Rightarrow eats(a, b)$
j_6 $\exists eats.(Sheep \sqcup Cow)(a) \Rightarrow (Sheep \sqcup Cow)(a)$
j_7 $(Animal \sqcap \forall eats.\neg Animal)(a) \Rightarrow Animal(a)$
j_8 $(Animal \sqcap \forall eats.\neg Animal)(a) \Rightarrow \forall eats.\neg Animal(a)$
j_9 $eats(a, b), \forall eats.\neg Animal(a) \Rightarrow \neg Animal(b)$
j_{10} $(Sheep \sqcup Cow)(a) \Rightarrow Sheep(a)$ (non-deterministic)
j_{11} $(Sheep \sqcup Cow)(a) \Rightarrow Cow(a)$ (non-deterministic)
j_{12} $Sheep(b), Sheep \sqsubseteq Animal \Rightarrow Animal(b)$
j_{13} $Animal(b), \neg Animal(b) \Rightarrow \bot$
j_{14} $Cow(b), Cow \sqsubseteq Animal \sqcap \forall eats.\neg Animal \Rightarrow (Animal \sqcap \forall eats.\neg Animal)(b)$
j_{15} $(Animal \sqcap \forall eats.\neg Animal)(b) \Rightarrow Animal(b)$
j_{16} $(Animal \sqcap \forall eats.\neg Animal)(b) \Rightarrow \forall eats.\neg Animal(b)$

and the justification graph is shown in Figure 2.

In a standard ATMS [11], each node has a *label* consisting of a set of *environments*. An environment is a minimal set of axioms from which the corresponding formula is

[1] Note that we use the term justification as it is used in ATMS literature, rather than to mean the minimal set of axioms responsible for an entailment as in, e.g., [4].

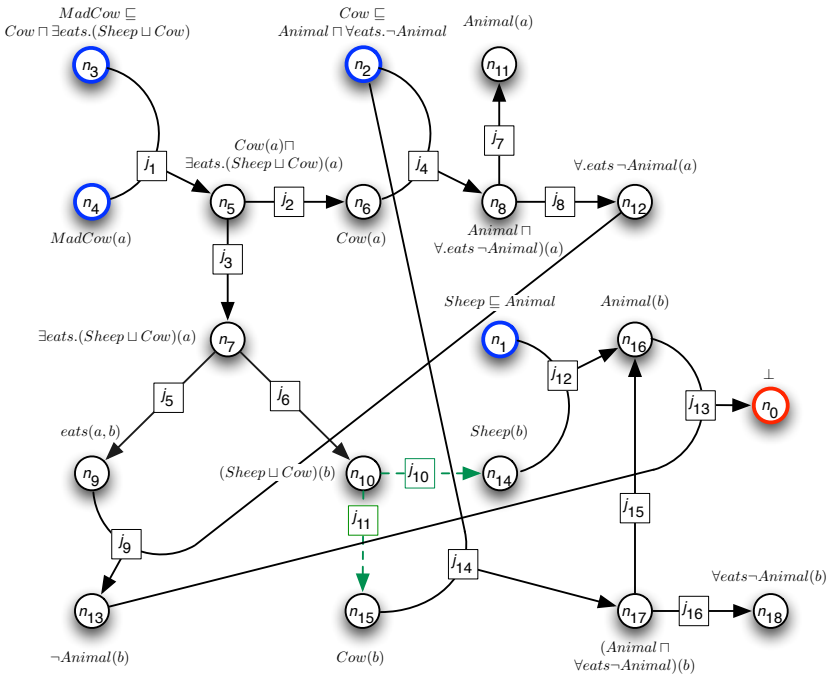

Fig. 2: Justification graph. Formula nodes are round, axioms are blue, $\bot$ is red. Justification nodes are square, non-deterministic justifications are green with dashed arrows.

derivable (an explanation). For example, $Animal(a)$ in Figure 2 would have an environment $\{Cow \sqsubseteq Animal \sqcap \forall eats.\neg Animal, MadCow \sqsubseteq Cow \sqcap \exists eats.(Sheep \sqcup Cow), MadCow(a)\}$. Labels are computed and minimised at the same time as the justification graph is built. In contrast, in AOD, labels are computed only for the nodes which belong to the part of the justification graph which is involved in the derivation of $\bot$ (is reachable from $\bot$ following the edges backwards), and only after the graph is complete. In our example, the relevant part is the graph without the justifications j_7, j_{16} and the nodes n_{11}, n_{18}.

4 Computing Labels

In this section, we explain how the standard ATMS label computation algorithms are generalised to deal with disjunctions. Basically, the generalisation consists in keeping track of dependencies on disjunctive choices in addition to dependencies on axioms.

As in a standard ATMS, each node in the justification graph has a *label* consisting of a set of environments. However in the D-ATMS, an environment represents a set of axioms and choices under which a particular formula holds.

Definition 1 (environment). *An environment e is a pair $(\mathcal{A}, \mathcal{C})$ where $\mathcal{A}$ is a set of axioms and $\mathcal{C}$ is a sequence of choice sets $[c_1, \ldots, c_k]$ of length $k \geq 0$. Each choice set*

c_i is a pair (d_i, b_i) where $d_i = \psi_1 \sqcup \ldots \sqcup \psi_n$ is a disjunction and $b_i \subset \{\psi_1, \ldots, \psi_n\}$ is a set of choices for d_i (i.e., a subset of the disjuncts appearing in the disjunction).

The presence of an environment $(\mathcal{A}, \mathcal{C})$ in the label of a node n_ϕ indicates that ϕ can be derived from the axioms $\mathcal{A}$ together with a sequence of choices from $\mathcal{C}$. The choice sequence corresponds to a (set of) tableau branch(es): each *choice* consists of a disjunction d_i and one or more of the disjuncts appearing in d_i. If ϕ can be derived from all the disjuncts appearing in d_i, we have eliminated dependency on all choices for d_i, and the choice set for d_i can be removed from $\mathcal{C}$. If the sequence of choice sets is empty, then ϕ does not depend on any choices (i.e., it can be derived from only the axioms $\mathcal{A}$). For example, the presence of the environment $(\{\phi_1, \ldots, \phi_k\}, [\,])$ in the label of a node n_ϕ means that ϕ has been derived by the reasoner from the axioms $\phi_1, \ldots, \phi_k$.

Environments in the D-ATMS thus capture the branching structure of a tableau. For example, in the tableau in Figure 1 an environment for $B_1(a)$ will have a choice sequence $[((B_1 \sqcup B_2 \sqcup B_3)(a), B_1(a))]$ and $C_1(b)$ will have a choice sequence $[((B_1 \sqcup B_2 \sqcup B_3)(a), B_1(a)),\ ((C_1 \sqcup C_2)(b), C_1(b))]$. The order of choice sets in a choice sequence comes from the order in which the $\sqcup$-rule is applied to disjunctions on the corresponding branch. If one choice sequence corresponds to a prefix of another, then the first choice sequence depends on fewer disjunctive choices. This intuition may be helpful when considering the definition of subsumption for environments below.

The *label* of a node contains the set of environments from which the formula corresponding to the node can be derived. The label of $n_\perp$ consists of a set of inconsistent environments or nogoods.

To define the D-ATMS algorithms for computing labels, we need the following primitive operations on environments and labels which generalise and extend the corresponding notions in [11].

We say that a choice sequence $\mathcal{C}_1$ is a prefix of a choice sequence $\mathcal{C}_2$, $\mathcal{C}_1 \preceq \mathcal{C}_2$, if $\mathcal{C}_1 = [(d_1, b_1), \ldots, (d_k, b_k)]$ and $\mathcal{C}_2 = [(d'_1, b'_1), \ldots, (d'_n, b'_n)]$, $k \leq n$ and for every $i \leq k$, $d_i = d'_i$ and $b'_i \subseteq b_i$. $\mathcal{C}_1 \prec \mathcal{C}_2$ iff $\mathcal{C}_1 \preceq \mathcal{C}_2$ and $\mathcal{C}_2 \not\preceq \mathcal{C}_1$.

Definition 2 (Subsumption of environments). *An environment $(\mathcal{A}_1, \mathcal{C}_1)$ subsumes an environment $(\mathcal{A}_2, \mathcal{C}_2)$, $(\mathcal{A}_1, \mathcal{C}_1) \subseteq_s (\mathcal{A}_2, \mathcal{C}_2)$ iff $\mathcal{A}_1 \subseteq \mathcal{A}_2$, and $\mathcal{C}_1 \preceq \mathcal{C}_2$. $(\mathcal{A}_1, \mathcal{C}_1) \subset_s (\mathcal{A}_2, \mathcal{C}_2)$ iff $(\mathcal{A}_1, \mathcal{C}_1) \subseteq_s (\mathcal{A}_2, \mathcal{C}_2)$ and $(\mathcal{A}_2, \mathcal{C}_2) \not\subseteq_s (\mathcal{A}_1, \mathcal{C}_1)$.*

An environment e is *nogood* if it is subsumed by an environment in the label of the false node $n_\perp$.

Definition 3 (Union of environments). *The union of two environments $e_1 = (\mathcal{A}_1, \mathcal{C}_1)$ and $e_2 = (\mathcal{A}_2, \mathcal{C}_2)$, $e_1 \cup_\preceq e_2 = (\mathcal{A}_1 \cup \mathcal{A}_2, \mathcal{C}_1 \cup_\preceq \mathcal{C}_2)$ if $\mathcal{C}_1$ and $\mathcal{C}_2$ are sequences of choice sets for which $\mathcal{C}_1 \cup_\preceq \mathcal{C}_2$ is defined, otherwise $e_1 \cup_\preceq e_2$ is not defined. $\cup_\preceq$ for sequences of choice sets is defined as follows:*

1. *if $\mathcal{C}_1 \preceq \mathcal{C}_2$ then $\mathcal{C}_1 \cup_\preceq \mathcal{C}_2 = \mathcal{C}_2$;*
2. *if $\mathcal{C}_2 \preceq \mathcal{C}_1$ then $\mathcal{C}_1 \cup_\preceq \mathcal{C}_2 = \mathcal{C}_1$;*
3. *for all other cases, $\mathcal{C}_1 \cup_\preceq \mathcal{C}_2$ is not defined.*

Intuitively, environments of two antecedents can be combined by $\cup_\preceq$ to form an environment of the consequent if the antecedents belong to the same branch of the tableau.

Definition 4 (Merge of environments). *The merge of two environments $e_1 = (\mathcal{A}_1, \mathcal{C}_1)$ and $e_2 = (\mathcal{A}_2, \mathcal{C}_2)$, $e_1 \cup_+ e_2 = (\mathcal{A}_1 \cup \mathcal{A}_2, \mathcal{C}_1 \cup_+ \mathcal{C}_2)$ if $\mathcal{C}_1$ and $\mathcal{C}_2$ are sequences of choice sets for which $\cup_+$ is defined. Otherwise, $e_1 \cup_+ e_2$ is not defined. $\cup_+$ for sequences of choice sets is defined as follows:*

1. *if $\mathcal{C}_1 = [(d_1, b_1), \ldots, (d_n, b_n)]$ and $\mathcal{C}_2 = [(d'_1, b'_1), \ldots, (d'_n, b'_n)]$, $n \geq 1$, and for every $i < n$ $d_i = d'_i$ and $b'_i = b_i$ (in other words, $\mathcal{C}_1$ and $\mathcal{C}_2$ are the same apart from their last element), $d_n = d'_n$, $b_n \neq b'_n$, then*
   (a) *if $b_n \cup b'_n$ does not include all the disjuncts in d_n, then $\mathcal{C}_1 \cup_+ \mathcal{C}_2 = [(d_1, b_1), \ldots, (d_n, b_n \cup b'_n)]$*
   (b) *$\mathcal{C}_1 \cup_+ \mathcal{C}_2 = [(d_1, b_1), \ldots, (d_{n-1}, b_{n-1})]$ otherwise;*
2. *for all other cases, $\mathcal{C}_1 \cup_+ \mathcal{C}_2$ is not defined.*

Intuitively, if the same formula belongs to all children of a disjunctive node in a tableau, then it can be lifted 'up' to the parent, otherwise, $\cup_+$ merges two subtrees into one subtree where the formula belongs to all children. Recall that the label of a node is the set of all environments from which the node can be derived.

Definition 5 (Union of labels). *The union of two labels L_1 and L_2, $L_1 \cup_+ L_2 = L_1 \cup L_2 \cup \{e_1 \cup_+ e_2 \mid e_1, e_2 \in L_1 \cup L_2\}$.*

We can now give a sketch of how labels are computed.

Given a justification graph as in Figure 2, we first compute the *justification closure* J for $n_\perp$, namely the set of justifications that have $n_\perp$ as a consequent, together with the justifications of the antecedents of those justifications, and so on until we reach justifications whose antecedents are axiom nodes. Initially, the labels of all nodes in J other than axiom nodes are empty, and the label of each axiom node in J contains a single environment consisting of the axiom itself.

The justifications in J are processed in order of their ids. For each justification $j : n_{\phi_1}, \ldots, n_{\phi_k} \Rightarrow n_\phi \in J$ in turn, if j is deterministic (corresponds to any inference rule apart from the $\sqcup$-rule), then for every k-tuple of environments from the labels of $n_{\phi_1}, \ldots, n_{\phi_k}$ (every way to derive the premises) we take their $\cup_\leq$ union (which means, we only combine derivations on the same branch), remove any of the resulting environments which are subsumed (to guarantee minimality), remove nogoods and, if the label of n_ϕ has changed as a result, propagate the changes to the nodes reachable by following already processed (having a smaller id) justification links from n_ϕ (since we discovered a new way to derive those formulas, too).

If a justification $j : n_d \Rightarrow n_{\psi_i}$ is non-deterministic (corresponds to $\sqcup$-rule), then we need to make sure that the choices corresponding to splitting d are added at the correct points in the tableau tree (recall Figure 1). New branches should be added under each existing branch in the tableau where the disjunction is derivable. To reflect this tableau structure in the label of n_{ψ_i}, for each environment $(\mathcal{A}, \mathcal{C})$ appearing in the label of n_d we compute the set of choice sequences of maximal length appearing in any label, $\{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$. Each element $\mathcal{C}_s$ $(1 \leq s \leq k)$ of this set is maximal (corresponds to a complete branch ending in a leaf), and $\mathcal{C}$ is a prefix of $\mathcal{C}_s$. For each such $\mathcal{C}_s$ we add an environment $(\mathcal{A}, \mathcal{C}_s + (d, \{\psi_i\}))$ to a set L which is a new set of environments for ψ_i generated by j. For example, in Figure 1, when the $\sqcup$-rule is applied to

$(C_1 \sqcup C_2)(b)$, the only choice sequence appearing in its label is [ ]. The set of choice sequences of maximal length which have [ ] as a prefix are $[((B_1 \sqcup B_2 \sqcup B_3)(a), B_1(a))]$, $[((B_1 \sqcup B_2 \sqcup B_3)(a), B_2(a))]$, and $[((B_1 \sqcup B_2 \sqcup B_3)(a), B_3(a))]$. The choice sequences in the environments of $C_1(b)$ and $C_2(b)$ become $[((B_1 \sqcup B_2 \sqcup B_3)(a), B_1(a)), ((C_1 \sqcup C_2)(a), C_1(a))]$, $[((B_1 \sqcup B_2 \sqcup B_3)(a), B_1(a)), ((C_1 \sqcup C_2)(a), C_2(a))]$, etc. Finally we add L to the old label of ψ_i using $\cup_+$, remove any subsumed environments and nogoods, and propagate the changes to the nodes reachable from ψ_i by following already processed justifications from ψ_i in J.

The label computation algorithms are correct, in that every set of axioms Γ' from which $\bot$ can be derived given the set of justifications produced by the reasoner is a superset of the axioms appearing in some environment in the label of $n_\bot$, and $\bot$ can be derived from every environment in its label.

5 Experimental Results

We have developed a prototype implementation of our approach.[2] Both the reasoner and the D-ATMS are implemented in Pop-11.[3] The tableaux reasoner is implemented as a set of six inference rules using Poprulebase, a Pop-11 rule interpreter.

To evaluate our approach, we performed experiments in which we compared the performance of our prototype system when providing all minimal explanations for inconsistencies in a variety of unfoldable $\mathcal{ALC}$ TBoxes with that of MUPSter [23] and Pellet [24] (version 2.2.2). We chose to compare the D-ATMS with MUPSter and Pellet as they represent different approaches to finding all minimal explanations for an inconsistency. Both use a glass-box approach (extending the reasoner with dependency tracking), but MUPSter finds all minimal explanations, while Pellet finds a single minimal explanation, which is then combined with Reiter's Hitting Set algorithm [19] to find all other explanations [9, 24]. (In our experiments, we used Pellet's glass-box approach, as this typically requires less time to find an explanation [9].) The experiments were performed on a Intel Dual Core 2.16GHz, 2GB RAM PC running Ubuntu. All times are CPU times in ms and represent the average of 5 runs. Only the time actually used for generating explanations is given. We do not count the time AOD, MUPSter, and Pellet spend parsing and loading the ontologies, nor the time required for them to render the explanations.

To test the correctness of our implementation, we compared the results for AOD with those of MUPSter on the set of 1,611 randomly generated unfoldable $\mathcal{ALC}$ TBoxes used by Schlobach to evaluate the performance of MUPSter [23].[4] For each ontology, we obtained a list of unsatisfiable concept names from RacerPro before finding all minimal explanations for each unsatisfiable concept name.[5] The explanations generated

[2] AOD is available at `http://www.agents.cs.nott.ac.uk/research/logics/ontologies`.

[3] `http://www.cs.bham.ac.uk/research/projects/poplog/freepoplog.html`

[4] The dataset is available at `http://www.few.vu.nl/~schlobac/software.html`.

[5] `http://www.racer-systems.com/products/racerpro`

by both systems were the same, apart from one case where MUPSter returned a non-minimal explanation.[6]

We also recorded the CPU time required for AOD, MUPSter and Pellet to generate explanations for each ontology. In one case MUPSter did not produce an explanation within 5000 seconds and the run was aborted. We omitted this case and the case in which MUPSter returned a non-minimal explanation from our analysis, and in the following we consider only the remaining 1609 cases. Overall, AOD was noticeably faster than both MUPSter and Pellet, with an average execution time of 30ms (median 9ms) compared to 1001ms (median 166ms) for MUPSter and 478ms (median 383) for Pellet.

To evaluate the performance of AOD on more realistic examples, we used the Geo ontology [23], the Biochemistry-primitive ontology from the TONES repository,[7] a fragment of the Ordnance Survey BuildingsAndPlaces ontology,[8] and the Adult Mouse Brain Ontology from the NCBO BioPortal.[9] The Biochemistry-primitive, BuildingsAndPlaces, and Adult Mouse Brain ontologies were translated into $\mathcal{ALC}$ by removing axioms for inverse roles and role inclusions. As in [23], the Geo ontology was made incoherent by adding disjointness axioms of the form $DJ(A_1, \ldots, A_n)$ stating that the concepts $A_1, \ldots, A_n$ are pairwise disjoint. To handle the disjointness axioms, we added the following inference rule to the reasoner:

dj**-rule** from $A_i(a)$ and $DJ(A_1, \ldots, A_n)$ derive $\neg A_j(a)$ for all $j \neq i, j \in \{1, \ldots, n\}$.

To make the other ontologies incoherent, we choose to systematically create unsatisfiable concepts from existing ontology entailments, allowing us to control the number of unsatisfiable concepts and the form of the resulting explanations. For each ontology, we randomly selected 10 pairs of concepts (A, B) where $A \sqsubseteq B$ is non-trivially entailed by the ontology, i.e., $A \sqsubseteq B \notin \mathcal{T}$. Then for each entailment, $A \sqsubseteq B$, we created a concept $EntailmentA\_B \sqsubseteq A \sqcap \neg B$. Finding all minimal explanations for the entailment $A \sqsubseteq B$ thus becomes equivalent to finding all minimal explanations for the unsatisfiability of $EntailmentA\_B$.

Table 1: Average execution times for AOD, MUPSter and Pellet.

Ontology	Axioms	Unsat concepts	AOD	MUPSter	Pellet
Geo	500	11	72	259	3649
Biochemistry-primitive	265	10	20	70	418
BuildingsAndPlaces	124	10	42	88	515
Adult Mouse Brain	3447	10	802	1381	3443

[6] For the TBox `tbox_50_6_1_1_3_5_v1` and unsatisfiable concept $A49$ MUPSter returns $\{A49, A37, A26, A34, A0\}$ as an explanation for the unsatisfiability of $A49$, while the D-ATMS returns $\{A49, A37, A34, A0\}$.

[7] `http://owl.cs.manchester.ac.uk/repository`

[8] `http://www.ordnancesurvey.co.uk/oswebsite/ontology/BuildingsAndPlaces/v1.1/BuildingsAndPlaces.owl`

[9] `http://bioportal.bioontology.org/ontologies/1290`

The results are presented in Table 1. The second and third columns show the number of axioms and the total number of unsatisfiable concepts in each ontology. As can be seen, AOD is 1.5 to 3.5 times faster than MUPSter and 4 to 50 times faster than Pellet on these ontologies.

6 Related Work

Two main approaches to axiom pinpointing have been proposed in the literature: glass-box methods and black-box methods. A glass-box method extends a description logic reasoner with some method of dependency tracking. A black-box method, e.g., [9], uses the reasoner as an oracle to determine whether a set of axioms results in an inconsistency or a concept is unsatisfiable with respect to a set of axioms, and then shrinks that set to find a minimal set of reasons for the inconsistency or concept unsatisfiability. Black-box methods have the advantage of being reasoner-independent. However it can be argued that glass-box methods provide additional information in the form of a derivation, which is useful for debugging, e.g., to present the user with a summary of the derivation and which parts of the axioms were used to derive a contradiction.

AOD adopts a glass-box approach to ontology debugging. To date, much of the work on glass box approaches, e.g., [22, 8, 16, 15], has been tailored to a particular logic. More recently, Baader and Peñaloza [3] have proposed a generic tableau rule specification format and a pinpointing algorithm that works for reasoners specified in this format. They also show that termination of a tableau reasoner for satisfiability does not necessarily lead to the termination of its pinpointing extension. In addition, for tableau reasoners that require a blocking condition for termination, e.g., full $\mathcal{ALC}$, it is not sufficient for the pinpointing extension to use the same blocking condition as the reasoner, because the pinpointing extension needs to take into account not only the presence of an assertion in $\mathcal{A}$, but also its justifications to determine if a tableau rule instance should be blocked. In [3] they give a characterisation of a class of terminating tableaux where the blocking condition yields a complete and terminating pinpointing extension. However, to the best of our knowledge, this approach has not been implemented. In [18] we sketched an approach to using an ATMS for ontology debugging in a description logic without disjunctions, but did not provide an implementation.

The ATMS as described in [11] does not support non-deterministic choices. However several approaches to handling disjunctions in an ATMS have been proposed in the literature. In [12] de Kleer extended the original ATMS to encode disjunctions of assumptions (axioms) by introducing a set of hyper-resolution rules. However, such rules may significantly reduce the efficiency of the ATMS. Another approach uses a justification for $\bot$ by negated assumptions to represent a disjunction of assumptions, e.g., $A \vee B$ can be encoded by the justification $\neg A, \neg B \Rightarrow \bot$ [13]. Both of these approaches are limited to encoding a disjunction of assumptions (axioms). However, in ontology debugging, disjunctions often appear in concept descriptions. In contrast, the D-ATMS allows disjunctions of nodes corresponding to arbitrary formulas. In [20] the original ATMS was generalised to a clause management system (CMS) where justifications are arbitrary disjunctive clauses. To find the 'minimal support' for a clause, the CMS implementation described in [14] uses a method for computing prime implicants which

relies on justifications being clauses consisting of literals to which the resolution rule can be applied. Adopting such an approach would require translating TBox axioms into clauses, and more importantly, finding some way of mapping the clauses returned by the CMS back to the original TBox. The latter in particular is a non-trivial problem. Label computation in the D-ATMS has some similarities with lazy label evaluation in assumption-based truth maintenance systems, e.g., [10], and the restriction to $n_\perp$ can be seen as a special case of focussing the ATMS, e.g., [6]. Such approaches have been shown to offer significant performance improvements relative to the ATMS described in [11].

7 Conclusion

We described AOD, a system for debugging unfoldable $\mathcal{ALC}$ TBoxes based on an ATMS with disjunctions. Our approach is correct and complete with respect to a reasoner for $\mathcal{ALC}$ with unfoldable TBoxes. We presented experimental results which suggest that its performance compares favourably with that of MUPSter and Pellet. As the D-ATMS maintains an explicit justification structure, it is straightforward to generate explanations of *how* a contradiction is derivable intended for human users — the D-ATMS essentially keeps intermediate steps in a derivation and can produce them on request.

We believe the D-ATMS is a promising new approach to ontology debugging. Although our approach was developed for $\mathcal{ALC}$ with unfoldable TBoxes, the reasoner and the reason maintenance component are only loosely coupled, and the D-ATMS can be adapted to work with other tableau reasoners. Characterising the conditions under which a terminating tableau algorithm can be combined with the D-ATMS to produce a debugging tool that will find all minimal explanations of $\perp$ is further work. One possible approach would be to build on the results of [3]. The production of more user-friendly explanations of how a contradiction is derivable is also a topic of future work.

References

1. Baader, F., Hollunder, B.: Embedding defaults into terminological representation systems. Journal of Automated Reasoning 14, 149–180 (1995)
2. Baader, F., Hollunder, B.: A terminological knowledge representation system with complete inference algorithms. In: Processing Declarative Knowledge, LNCS, vol. 567, pp. 67–86 (1991)
3. Baader, F., Peñaloza, R.: Axiom pinpointing in general tableaux. Journal of Logic and Computation 20(1), 5–34 (2010)
4. Bail, S., Horridge, M., Parsia, B., Sattler, U.: The justificatory structure of the NCBO BioPortal ontologies. In: Proceedings of the 10th International Semantic Web Conference (ISWC 2011). LNCS, vol. 7031, pp. 67–82. Springer (2011)
5. Broekstra, J., Kampman, A.: Inferencing and truth maintenance in RDF schema. In: Proceedings of the First International Workshop on Practical and Scalable Semantic Systems. CEUR Workshop Proceedings, vol. 89. CEUR-WS.org (2003)
6. Forbus, K.D., de Kleer, J.: Focusing the ATMS. In: Proceedings of the Seventh National Conference on Artificial Intelligence. pp. 193–198. AAAI Press/MIT Press (1988)

7. Guo, Y., Heflin, J.: An initial investigation into querying an untrustworthy and inconsistent web. In: Proceedings of the ISWC'04 Workshop on Trust, Security, and Reputation on the Semantic Web. vol. 127. CEUR-WS.org (2004)
8. Kalyanpur, A., Parsia, B., Sirin, E., Hendler, J.: Debugging unsatisfiable classes in OWL ontologies. Journal of Web Semantics 3(4), 268–293 (2005)
9. Kalyanpur, A., Parsia, B., Horridge, M., Sirin, E.: Finding all justifications of OWL DL entailments. The Semantic Web pp. 267–280 (2008)
10. Kelleher, G., van der Gaag, L.: The LazyRMS: Avoiding work in the ATMS. Computational Intelligence 9(3), 239–253 (1993)
11. de Kleer, J.: An assumption-based TMS. Artificial Intelligence 28(2), 127–162 (1986)
12. de Kleer, J.: Extending the ATMS. Artificial Intelligence 28(2), 163–196 (1986)
13. de Kleer, J.: A general labeling algorithm for assumption-based truth maintenance. In: Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI'88). pp. 188–192. AAAI Press/MIT Press (1988)
14. de Kleer, J.: An improved incremental algorithm for generating prime implicates. In: Proc. of the Tenth National Conference on Artificial Intelligence (AAAI'92). pp. 780–785. AAAI Press/MIT Press (1992)
15. Lam, J.S.C., Sleeman, D.H., Pan, J.Z., Vasconcelos, W.W.: A fine-grained approach to resolving unsatisfiable ontologies. Journal of Data Semantics 10, 62–95 (2008)
16. Meyer, T.A., Lee, K., Booth, R., Pan, J.Z.: Finding maximally satisfiable terminologies for the description logic ALC. In: Proceedings of the Twenty First National Conference on Artificial Intelligence (AAAI'06) (2006)
17. Nebel, B.: Terminological reasoning is inherently intractable. Artificial Intelligence 43(2), 235–249 (1990)
18. Nguyen, H., Alechina, N., Logan, B.: Ontology debugging with truth maintenance systems. In: Bundy, A., Lehmann, J., Qi, G., Varzinczak, I.J. (eds.) ECAI-10 Workshop on Automated Reasoning about Context and Ontology Evolution (ARCOE-10), Workshop Notes. pp. 13–14. Lisbon, Portugal (2010)
19. Reiter, R.: A theory of diagnosis from first principles. Artificial Intelligence 32(1), 57–95 (1987)
20. Reiter, R., de Kleer, J.: Foundations of assumption-based truth maintenance systems: Preliminary report. In: Proceedings of the Sixth National Conference on Artificial Intelligence, (AAAI'87). pp. 183–189 (1987)
21. Ren, Y., Pan, J.Z.: Optimising ontology stream reasoning with truth maintenance system. In: Proceedings of the ACM Conference on Information and Knowledge Management (CIKM 2011 (2011)
22. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: Gottlob, G., Walsh, T. (eds.) Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03). pp. 355–360. Morgan Kaufmann (2003)
23. Schlobach, S., Huang, Z., Cornet, R., van Harmelen, F.: Debugging incoherent terminologies. Journal of Automated Reasoning 39(3), 317–349 (2007)
24. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Journal of Web Semantics 5(2), 51–53 (2007)