International Journal of Software and Informatics, ISSN 1673-7288
http://www.ijsi.org, ijsi@iscas.ac.cn, +86-10-62661048
IJSI, 2023, 13(3): 359–374, doi: 10.21655/ijsi.1673-7288.00304

International Journal of Software and Informatics

Research Article

Consequence-based Axiom Pinpointing for Expressive Description Logic Ontologies

Jing Li (李静)[1,2], Dantong Ouyang (欧阳丹彤)[1,2], Yuxin Ye (叶育鑫)[1,2]

[1] (College of Computer Science and Technology, Jilin University, Changchun 130012, China)
[2] (Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education (Jilin University), Changchun 130012, China)
Corresponding author: Yuxin Ye, yeyx@jlu.edu.cn

Abstract Axiom pinpointing has attracted extensive interest in Description Logics (DLs) due to its effect of exploring explicable defects in the DL ontology and searching for hidden justifications for logical implication. Balancing the expressive power of DLs and the solving efficiency of reasoners has always been the focus of axiom pinpointing research. This study, from both glass-box and black-box perspectives, proposes a consequence-based method for axiom pinpointing. The glass-box method uses modified consequence-based rules (pinpointing rules) to trace the specific process of inference and introduces the concept of pinpointing formula to establish the correspondence between the label of the Boolean formula and all the minimal axiom sets. The black-box method directly calls the reasoner based on the unmodified consequence-based rules and further uses the Hitting Set Tree (HST) to compute all justifications of logical implication. Finally, a reasoning tool is designed based on the two axiom pinpointing algorithms for expressive DL ontologies. Its feasibility is verified theoretically and experimentally, and its solving efficiency is compared with that of existing axiom pinpointing tools.

Keywords axiom pinpointing; consequence-based method; pinpointing rules; glass-box and black-box

Citation Li J, Ouyang DT, Ye YX. Consequence-based axiom pinpointing for expressive description logic ontologies, *International Journal of Software and Informatics*, 2023, 13(3): 359–374. http://www.ijsi.org/1673-7288/304.htm

Description Logics (DLs) are common formalized languages for knowledge representation, and the ontology knowledge base represented by this languages includes TBoxes and ABoxes[1]. Different constructors in DLs indicate different abilities to represent an ontology. The ontology that needs to be constructed becomes more and more complex with the expansion of its application, making it an inevitable trend from the study of unexpressive DL ontologies to that of expressive DL ontologies. Inference as an important content of ontology research can derive tacit knowledge from explicitly given knowledge. Axiom pinpointing is a special reasoning task proposed by Baader *et al*.[1], which mainly aims to find the Minimal Axiom sets (MinAs) of a given logical implication in an ontology and is commonly used to search for defects and hidden reasons for logical errors in the ontology. For example, in the biomedical

This is the English version of the Chinese article “强表达描述逻辑本体的后继式公理定位研究. 软件学报, 2023, 34(8): 3574–3586. DOI: 10.13328/j.cnki.jos.006870”
Funding items: National Natural Science Foundation of China (42050103, 62076108)
Received 2022-09-06; Revised 2022-10-13; Accepted 2022-12-14; IJSI published online 2023-09-27

ontology Snomed-CT[2], the relationship that "the amputation of a finger is the amputation of an arm" can be obtained based on the inference of the ontology, which means that the arm of a patient who receives a finger amputation must also be amputated. Such an inference is utterly absurd. Therefore, it is necessary to calculate the minimal logical reason for a given axiom from the ontology. Errors are inevitable in the construction, updating, merging, and other processes of ontologies, but developers or users of an ontology often have difficulty in understanding why a conclusion is true and thus feel even harder to decide how to modify the ontology without a conclusion. Therefore, the axiom pinpointing of logical implication has important practical significance and has attracted extensive interest from researchers.

Schlobach *et al*.[3] proposed the Tableau-based glass-box pinpointing method and modified the existing decision algorithm so that the justification of logical implication could be calculated from the running process of an algorithm. They introduced the concept of Minimal Unsatisfiability-Preserving Sub-TBoxes (MUPS), which, however, was only applied to acyclic $\mathcal{ALC}$ inconsistent ontologies. To calculate the justification of more expressive ontologies, Parsia *et al*.[4] came up with a glass-box method suitable for the $\mathcal{SHIF}(\mathcal{D})$ ontology, and Kalyanpur *et al*.[5] further extended it to the $\mathcal{SHOIN}(\mathcal{D})$ ontology. Ouyang *et al*.[6] raised the concept R-MUPS based on the ordered label calculus, which could effectively determine the covering characteristics between MUPS of unsatisfiable concepts to speed up the solving of MUPS. These methods are all Tableau-based glass-box pinpointing algorithms. Schlobach *et al*.[7] also put forward a black-box pinpointing method, which computed the justification by repeatedly calling an unmodified reasoner. Ji *et al*.[8] calculated the justification of implication using the relevance-based method. Zhang *et al*.[9] proposed a debugging and repairing strategy based on the clash path to reduce the scale of the debugging target to improve efficiency; they also put forward the definition of increment ontology-defined sequence to improve the expansion and contraction phase of the black-box method to improve the solving efficiency[10]. Gao *et al*.[11] explored the duality between MUPS and MCPS, designed a pinpointing method, and applied the parallel strategy, which also improved the efficiency.

To balance the reasoning ability and the expressive power of DLs, Baader *et al*.[12] studied the inference of $\mathcal{EL}$ languages, and they improved the inference efficiency by reducing the expressive power of the ontology. They encoded the propositional logic for the classification process of $\mathcal{EL}$ ontologies and obtained the justification of logical implication by solving the Minimal Unsatisfiable Subformula (MUS). Considering that the lightweight DL ontologies tend to be large, Gao *et al*.[13] put forward an ontology justification solving strategy based on an approximate kernel method so that the local search algorithm showed better performance. Ye *et al*.[14], by using the concept of key axioms, modified the contraction phase to avoid unnecessary detection of the selection function, thus improving the efficiency of solving one justification. These studies, in fact, borrow the black-box method of a high-performance SAT solver, which displays high solving efficiency in large-scale biomedical ontologies and yet is unapplicable to more expressive ontologies.

The difficulty in dealing with expressive DL ontologies of high complexity often greatly affects the solving efficiency of their axiom pinpointing. To improve the solving efficiency of axiom pinpointing, this study proposes a consequence-based axiom pinpointing method for expressive DL ontologies. Combining the Tableau-based method and the coding system in the $\mathcal{EL}$ ontology pinpointing method, this method transforms the DL expression into a multi-type clause equality logic expression by a series of transformation rules, which applies to more expressive ontologies. The consequence-based method was first applied to $\mathcal{EL}$ ontologies[15] and later to $\mathcal{ALCHI}$ ontologies[16]. Afterwards, Horrocks *et al*.[17] introduced the concept of context and developed a unified reasoning framework to make such a consequence-based method

applicable to the more expressive $\mathcal{ALCHI}$ ontologies. This reasoning framework has been extended to all DL languages except for data types by Horrocks *et al*.[18] using the pay-as-you-go method. Compared with the existing work, the innovations of this study are as follows. (1) Multi-type clause equality logic coding is used to extend the consequence-based axiom pinpointing for $\mathcal{ALC}$ ontologies proposed by Rafael *et al*.[19] to the consequence-based axiom pinpointing for expressive $\mathcal{ALCHIQ}$ ontologies. (2) Compared with the existing consequence-based axiom pinpointing theory proof work[19], this study realizes for the first time the consequence-based glass-box axiom pinpointing reasoner, completing the whole process from theory to practice and experimental test. (3) Based on the Tableau-based and SAT-based axiom pinpointing work, this study proposes a consequence-based black-box test method, which improves the consequence-based axiom pinpointing method system. The framework is shown in Figure 1.

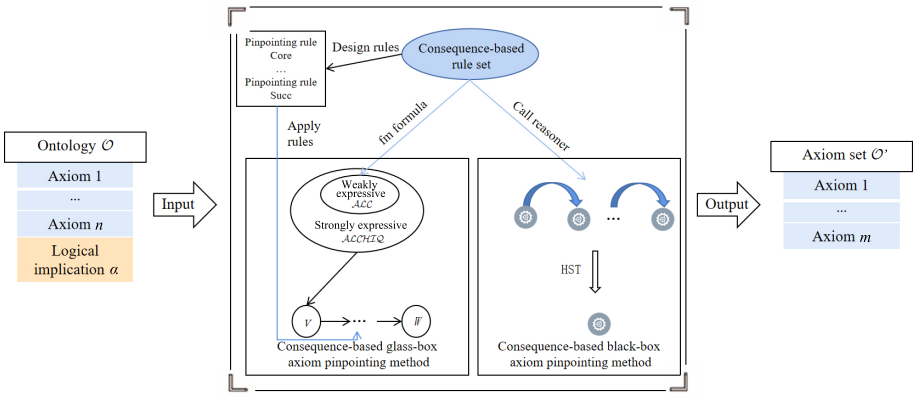

Figure 1 Framework of consequence-based axiom pinpointing method system

In this study, Section 1 introduces the basic knowledge required, including DLs, their equations, and transformation rules between DL axioms and DL clauses, as well as relevant concepts of the consequence-based method. Sections 2 and 3 describe the consequence-based glass-box pinpointing algorithm and consequence-based black-box pinpointing algorithm adopted in this study, respectively. Section 4 verifies the feasibility and effectiveness of the two pinpointing algorithms through comparative experiments with other reasoners. Finally, a summary is given.

1 Basic Knowledge

1.1 DL expression

DL is composed of four basic elements: concept, role, individual, and constructor, of which the number of constructors determines the expressive power of DLs, and more constructors indicate the higher expressive power of DLs. The grammar of DLs is defined by logical operators. The expressive description logic $\mathcal{ALCHIQ}$ is composed of γ (universal set), $\perp$ (empty set), A (atomic concept), $\neg A$ (atomic negative concept), $\cap$ (conceptual conjunction), $\cup$ (conceptual disjunction), n (number restriction), $\exists r.A$ (existential qualification), and $\forall r.A$ (universal qualification), and it allows the presence of hierarchical relations and inverse roles between roles. The semantics of DLs are defined by the interpretation $\mathcal{I}$; $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is an interpretation composed of the interpretive domain $\Delta^{\mathcal{I}}$ and the interpretive function $\cdot^{\mathcal{I}}$, of which $\Delta^{\mathcal{I}}$ is a non-empty set representing all objects in a domain, and the interpretive function $\cdot^{\mathcal{I}}$ maps the concept A to $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, a subset of $\Delta^{\mathcal{I}}$, and the role R to $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, a

binary relation on $\Delta^{\mathcal{I}}$.

The DL expression can be transformed into a multi-type clause equality logic expression in a way that preserves satisfiability and implication relations, and the multi-type signature is $\langle \Sigma^S, \Sigma^F \rangle$, where Σ^S is a non-empty set of types and Σ^F is a denumerable set of functional symbols. The equality $s \approx t$ is an expression consisting of terms s and t of the same type. In a multi-type clause equality logic, the literal is an equality or inequality, and the clause is a statement of the form $\forall \vec{x}(\Gamma \to \Delta)$, where the body Γ is a conjunction form of equalities, and the head Δ is a disjunction form of literals. $\forall \vec{x}$ represents all variables in this clause, and universal quantifiers are often omitted here. The strict (incomplete) order $\succ$ based on a non-empty set $\mathcal{S}$ is a non-reflexive and transitive binary relation between elements in $\mathcal{S}$, and it includes the non-strict order $\succeq$ via the reflexive closure of the strict order $\succ$.

The DL expression uses a dual-type DL signature $\langle a, p \rangle$, where type a represents the standard first-order logical term, and type p represents the standard first-order logical atom. A term of the form $f_j(t)$ is the f_j successor of t, while t is its precursor. A variable in the DL signature is $\{x\} \cup \{z_i\}_{i \geq 1}$, of type a, where x is called a central variable, and z_i is called an adjacent variable. DL-a-term is a term of the form z_i, x, or $f_i(x)$, and a DL-p-term is a term of the form $B_i(z_j)$, $B_i(x)$, $B_i(f_j(x))$, $S_i(z_j, x)$, $S_i(x, z_j)$, $S_i(x, f_j(x))$, or $S_i(f_j(x), x)$. The DL literal is an equality or A of the form $A \approx \mathit{true}$, where A is a DL-p-term or an equality or inequality composed of two DL-a-terms. The body Δ of the DL clause $\forall \vec{x}(\Gamma \to \Delta)$ can only be a term of the form $B_i(x)$, $S_i(z_j, x)$, or $S_i(x, z_j)$, and the head Γ of the DL clause can only be a DL literal.

O is an $\mathcal{ALCHIQ}$ ontology. The axioms for the normalized O are shown on the left side of Table 1, and the DL clauses obtained from the axioms of the ontology according to the above multi-type clause equality logic[20] are shown on the right side of Table 1.

Table 1 Transformation rule from axioms for the $\mathcal{ALCHIQ}$ ontology to DL clauses

No.	Transformation from ontology axioms to DL clauses			
DL1	$\sqcap_{1 \leq i \leq n} B_i \sqsubseteq \sqcup_{n+1 \leq i \leq m} B_i$	$\rightsquigarrow$	$\wedge_{1 \leq i \leq n} B_i(x) \to \vee_{n+1 \leq i \leq m} B_i(x)$	
DL2	$B_1 \sqsubseteq \geq nS.B_2$	$\rightsquigarrow$	$B_1(x) \to S(x, f_i(x))$	for $1 \leq i \leq n$
		$\rightsquigarrow$	$B_1(x) \to B_2(f_i(x))$	for $1 \leq i \leq n$
		$\rightsquigarrow$	$B_1(x) \to f_i(x) \not\approx f_j(x)$	for $1 \leq i < j \leq n$
DL3	$\exists S.B_1 \sqsubseteq B_2$	$\rightsquigarrow$	$S(z_1, x) \wedge B_1(x) \to B_2(z_1)$	
DL4	$B_1 \sqsubseteq \leq nS.B_2$	$\rightsquigarrow$	$S(z_1, x) \wedge B_2(x) \to S_{B_2}(z_1, x)$	
		$\rightsquigarrow$	$B_1(x) \wedge \wedge_{1 \leq i \leq n+1} S_{B_2}(x, z_i) \to \vee_{1 \leq i < j \leq n+1} z_i \approx z_j$	for fresh S_{B_2}
DL5	$S_1 \sqsubseteq S_2$	$\rightsquigarrow$	$S_1(z_1, x) \to S_2(z_1, x)$	
DL6	$S_1 \sqsubseteq S_2^-$	$\rightsquigarrow$	$S_1(z_1, x) \to S_2(x, z_1)$	

1.2 Consequence-based method

An arbitrary $\mathcal{ALCHIQ}$ ontology O is given, and it is necessary to judge whether $O \vDash Q$ is true, where $Q = \Gamma_Q \to \Delta_Q$ is a query clause. According to the consequence-based method proposed by Bate *et al.*[20], the logical implication derived from the ontology is represented as a context clause. Different from the definition of DL clause above, the context clause allows only variables x and y with special meaning, where the variable y corresponds to the precursor of x. The context structure is a directed graph, whose vertexes (also known as context) represent the set of domain elements in the input ontology model and whose edges are marked with successor function symbols. The context structure assigns each vertex v the following information:

(1) core_v, the core of the context v, determines the set of domain elements represented by v, which contains only two variables x and y.

(2) S_v, a set of context clauses containing no constants, allows only terms of the form x, y, and $f(x)$. Each context clause $\Gamma \rightarrow \Delta$ represents a logical implication of the ontology, and $core_v \subseteq \Gamma$. In brief, the context clause can be written as $\Gamma' \rightarrow \Delta$, where $\Gamma = core_v \wedge \Gamma'$.

(3) The context v is parameterized using reasoning rules[20] to select the order $\succ_v$ on which the literal and context clauses depend.

Edges in the context structure are marked with successor function symbols, and the edge marked with the function symbol f from context v to w can represent that if $core_w \neq \{\}$, then each element in v has an element in w as its f successor. To judge whether $O \vDash Q$ is true, the consequence-based method needs to construct a context structure, whose context represents all instances of Γ_Q in a model of the ontology O, and reasoning rules are applied to this context structure until the rule application reaches the saturation point, that is when reasoning rules can no longer be applied, it is necessary to query whether there is the clause $\bot \rightarrow \Delta_Q$.

2 Consequence-based Glass-box Pinpointing Algorithm

2.1 Axiom pinpointing problem

An ontology O and an axiom α are given. If $O \vDash \alpha$, then the axiom α is a logical implication of the ontology O. The main goal of axiom pinpointing is to find MinAs associated with a given logical implication, i.e., find an axiom set O' that meets $O' \subseteq O$ to make $O' \vDash \alpha$ true.

If each axiom in the ontology is associated with a unique propositional variable $lab(\alpha)$, $lab(O)$ represents the set of all propositional variables corresponding to the axioms in the ontology O. For example, the ontology O_1 has four axioms: $C \sqsubseteq \exists r.C, \exists r.C \sqsubseteq D, C \sqsubseteq \forall r \cdot D$, and $C \sqcap D \sqsubseteq \bot$. Four different propositional variables $ax1$, $ax2$, $ax3$, and $ax4$ are used as the labels of the four axioms, thus $lab(O_1) = \{ax1, ax2, ax3, ax4\}$. The propositional assignment is used to represent a set of true variables, and assignment V and the axiom set O, the V-projection of the O is the set $O_V := \{\alpha \in O \mid lab(\alpha) \in V\}$. For example, given the assignment $V_1 = \{ax1, ax3\}$ of the ontology O_1 , its V_1-projection is the set $O_{V_1} := \{C \sqsubseteq \exists r.C, C \sqsubseteq \forall r.D\}$. A monotone Boolean formula φ based on $lab(O)$ is a Boolean formula that uses only variables in $lab(O)$, disjunctions ($\wedge$), and conjunctions ($\vee$). Given an axiom set O and an axiom α, a monotone Boolean formula φ based on $lab(O)$ is called a pinpointing formula if, for all assignments $V \subseteq lab(O)$, $O \vDash \alpha$if and only if V meets φ. The axiom set corresponding to the pinpointing formula φ represents all MinAs associated with $O \vDash \alpha$. For example, the pinpointing formula of $O_1 \vDash C \sqsubseteq \bot$ is $\varphi_1 = ax_1 \wedge ax_4 \wedge (ax_2 \vee ax_3)$, since for all assignments meeting $V_1 \subseteq lab(O_1)$, $O_1 \vDash C \sqsubseteq \bot$ is true if and only if V_1 meets φ_1; thus the axiom sets $\{ax1, ax2, ax4\}$ and $\{ax1, ax3, ax4\}$ corresponding to the pinpointing formula φ_1 represent all MinAs of $O_1 \vDash C \sqsubseteq \bot$.

2.2 Consequence-based pinpointing rules

Rafael *et al.*[19] proposed the consequence-based reasoning rules of weakly expressive DL ontologies, and such a simple rule set fails to complete the reasoning task when the DL ontology extends from the weakly expressive $\mathcal{ALC}$ ontology to the strongly expressive $\mathcal{ALCHIQ}$ ontology. However, researchers in Ref. [18] encoded the DL ontology based on the multi-type clause equality logic for the consequence-based method of the $\mathcal{ALCHIQ}$ ontology and introduced the concept of context to develop a unified reasoning framework. In the extension method mentioned in Ref. [19], it is only necessary to decompose the rule into Left Hand Side (LHS) and Right Hand Side (RHS), and each time a reasoning rule is applied in the corresponding decision algorithm, the axioms contained in the LHS of the rule are added to the labels of the axioms obtained in the RHS during the pinpointing process, and then the fm formula is used to

ensure that the labels of the logical implication solved are part of its pinpointing formula. When the DL ontology becomes more expressive, the reasoning rules used in the decision algorithm of the $\mathcal{ALCHIQ}$ ontology include not only the axioms corresponding to the RHS added according to the LHS of the rule but also the deletion rule for avoiding redundancy and duplication, and the Succ and Pred rules used to extend the context structure in the directed graph. This requires the separate consideration of the extension of different rules under the reasoning framework, rather than using the fm formula directly as shown in Ref. [19]. How to design and prove the pinpointing rule set of extension is the core content of the glass-box pinpointing algorithm proposed in this study.

In Ref. [18], researchers encoded the DL ontology based on the multi-type clause equality logic for the consequence-based method of the $\mathcal{ALCHIQ}$ ontology and introduced the concept of context to develop a unified reasoning framework. First, each DL clause obtained by the axiom transformation rules was marked with a unique propositional variable, and the context clause of each implication α of the ontology could be marked with a monotonic Boolean formula φ_α. The implication α, is represented with $\alpha{:}\varphi_\alpha$, where the former part was the context clause of the implication α, and the latter part was the Boolean formula label of the clause. Then, the following eight pinpointing rules modified from consequence-based reasoning rules were given to trace the specific process of reasoning.

- Pinpointing rule Core: for each atom A in $core_v$, add a context clause $\bot \to A{:}\varphi_\alpha, \varphi_\alpha := \bot$ to the clause set S_v of v.
- Pinpointing rule Hyper: if $\wedge_{i=1}^{n} A_i \to \Delta{:}\varphi_\alpha \in O, \Gamma_i \to \Delta_i \vee A_i\sigma{:}\varphi_{\beta_i} \in S_v$, and there is a replacement σ such that $\sigma(x) = x$, $\Delta_i \not\succeq_v A_i\sigma$, $1 \le i \le n$, then add the new clause $\wedge_{i=1}^{n}\Gamma_i \to \vee_{i=1}^{n}\Delta_i \vee \Delta\sigma{:}\varphi_\gamma, \varphi_\gamma{:} = \varphi_\alpha \wedge \wedge_{i=1}^{n}\varphi_{\beta_i}$ to S_v.
- Pinpointing rules Eq: if $\Gamma_1 \to \Delta_1 \vee s_1 \approx t_1{:}\varphi_\alpha \in S_v, t_1 \not\succeq_v s_1, \Delta_1 \not\succeq_v s_1 \approx t_1$, and $\Gamma_2 \to \Delta_2 \vee s_2 \otimes t_2{:}\varphi_\beta \in S_v, \otimes \in \{\approx, \not\approx\}, s_2 \succ_v t_2, \Delta_2 \not\succeq_v s_2 \otimes t_2, s_2 \not\succeq_p = s_1$, then add a new clause $\Gamma_1 \wedge \Gamma_2 \to \Delta_1 \vee \Delta_2 \vee s_2[t_1]_p \otimes t_2{:}\varphi_\gamma, \varphi_\gamma := \varphi_\alpha \wedge \varphi_\beta$ to S_v.
- Pinpointing rule Ineq: if $\Gamma \to \Delta \vee t \not\approx t{:}\varphi_\alpha \in S_v$, then add a new clause $\Gamma \to \Delta{:}\varphi_\beta$, $\varphi_\beta := \varphi_\alpha$ to S_v.
- Pinpointing rule Factor: if $\Gamma \to \Delta \vee s \approx t \vee s \approx t'{:}\varphi_\alpha \in S_v, \Delta \cup \{s \approx t\} \not\succeq_v s \approx t', s \succ_v t'$, then add a new clause $\Gamma \to \Delta \vee t \not\approx t' \vee s \approx t'{:}\varphi_\beta, \varphi_\beta := \varphi_\alpha$ to S_v.
- Pinpointing rule Elim: if $\Gamma \to \Delta{:}\varphi_\alpha \in S_v, \Gamma \to \Delta{:}\varphi_\beta \in S_v \backslash (\Gamma \to \Delta)$, then delete the clause $\Gamma \to \Delta$, $\varphi_\beta := \varphi_\alpha \vee \varphi_\beta$ from S_v.
- Pinpointing rule Pred: if $\langle u, v, f\rangle \in \varepsilon, \wedge_{i=1}^{n} A_i \to \vee_{i=l+1}^{l+n} A_i{:}\varphi_\alpha \in S_v$, and $\Gamma_i \to \Delta_i \vee A_i\sigma{:}\varphi_{\beta_i} \in S_u$, $\Delta_i \not\succeq_u A_i\sigma$, $1 \le i \le l$, where $A_i \in \mathrm{Pr}(O)$[19], for each $l+1 \le i \le l+n$, then add the new clause $\wedge_{i=1}^{l}\Gamma_i \to \vee_{i=1}^{l}\Delta_i \vee \vee_{i=l+1}^{l+n} A_i\sigma{:}\varphi_\gamma, \sigma = \{x \mapsto f(x), y \mapsto x\}, \varphi_\gamma := \varphi_\alpha \wedge \wedge_{i=1}^{l+n}\varphi_{\beta_i}$ to the clause set S_u of u.
- Pinpointing rule Succ: if $\Gamma \to \Delta \vee A{:}\varphi_\alpha \in S_u, \Delta \not\succeq_u A$, where A contains $f(x)$, then let $\langle v, core', \succ'\rangle = strategy(f, K_1, D)$, and if $v \notin V$, let $\succ_v := \succ_v \cap \succ'$; if $v \notin V$, let $V := V \cup \{v\}$, $core_v := core'$, $\succ_v := \succ'$, $S_v := \varnothing$. Add a new edge $\langle u, v, f\rangle$ to ε, and if $A' \in K_2 \backslash core'_v$, add a new clause $A' \to A'{:}\varphi_{\beta_i}$ to S_v, where $\sigma = \{y \mapsto x, x \mapsto f(x)\}, K_1 = \{A' \in Su(O)^{[19]} | \top \to A'\sigma \in Su(O)\}, K_2 = \{A' \in Su(O) | \Gamma' \to \Delta' \vee A'\sigma \in Su(O)\}, \Delta' \not\succeq_u A'\sigma, \varphi_{\beta_i} := \varphi_\alpha$.

Different from the simple extension in Ref. [19], when the pinpointing rule Core adds a context clause to a clause set by the reasoning rule Core, the label $\varphi_\alpha := \bot$. is added to this clause. When the pinpointing rules Hyper and Eq and the rule Pred obtain a new axiom from several axioms by the corresponding reasoning rule, the disjunction form of those axiom labels is used to represent the label of the new axiom. When the pinpointing rule Ineq obtains a new axiom β from the axiom α by the reasoning rule Ineq, the label of α is used to represent that of

β. There are two special pinpointing rules, namely Elim and Succ. The Elim reasoning rule will delete the redundant axiom α according to axioms α and β, and then the label formula of the axiom β needs to be modified. $\varphi_\beta := \varphi_\alpha \vee \varphi_\beta$ is represented with a disjunction form to show that the preserved axiom β can be obtained either by the original reasoning process or by the process of obtaining the axiom α. The pinpointing rule Succ constructs a new context structure by the extension strategy, and a new clause is added to the clause set of this context structure, with its label formula from a clause that meets the reasoning rule in the extended original context structure.

The rule set constructed based on the above eight pinpointing rules can not only solve the axiom pinpointing problem of strongly expressive DL ontologies but also apply to weakly expressive DL ontologies. For example, for the $\mathcal{ALC}$ ontology, the pinpointing method mentioned in Ref. [19] requires no encoding of the ontology but directly applies eight consequence-based reasoning rules to the ontology in a pinpointing state to solve the justifications. In contrast, for the method proposed in this study, it is easier to match the pinpointing rule conditions for the clause set after the ontology is encoded since the constructor in the ontology is simple, and the pinpointing formula of the logical implication can be obtained when the algorithm terminates.

2.3 Consequence-based pinpointing algorithm and proof

The consequence-based glass-box pinpointing algorithm adds a marker to each clause in the DL clause set obtained by encoding the ontology O to obtain the pinpointing clause set O^{Pin}, adds markers to all clauses in the clause set of the context v to obtain the pinpointing clause set S_v^{Pin}, and adds markers to all clauses in the context structure D to obtain the context pinpointing structure D^{Pin}. The above pinpointing rules are applied to O^{Pin} and D^{Pin}; when no pinpointing rule can be applied, it can be said that the current context pinpointing structure D^{Pin} reaches the saturation point. The specific process of the algorithm is shown in Algorithm 1.

Algorithm 1. To solve the consequence-based glass-box pinpointing algorithm of all justifications of a given logical implication.

Input: axiom set O, axiom Q;
Output: all MinAs associated with $O \vDash Q$
Step 1. Construct an empty context structure D and select an extension strategy[20] K;
Step 2. Introduce a context q in D, where $core_q = \Gamma_Q$, and add markers for all axioms and logical implications;
Step 3. Apply the above pinpointing rules to D^{Pin} and O^{Pin}, until no rules can be applied and all rules are saturated;
Step 4. $O \vDash Q$ holds if and only if $\Gamma_Q \to \Delta_Q{:}\varphi_Q \in S_v^{\text{Pin}}$, and the Boolean formula marker φ_Q of this clause is its pinpointing formula;
Step 5. Return all MinAs corresponding to φ_Q;

If a reasoning rule can be applied in a state in the consequence-based method, the corresponding pinpointing rule of this rule can still be applied in the pinpointing algorithm. In other words, the application of a rule in the decision process is the same as that in the axiom pinpointing algorithm. The difference is that the decision algorithm changes the context structure and deletes or adds a new clause in the clause set, while the pinpointing algorithm traces the specific process of reasoning by the pinpointing rule and modifies the monotonic Boolean formula of the clause deleted or added during the reasoning process. Yet, all logical implications produced by pinpointing rules can also be produced by reasoning rules.

A is used to represent the current state of the decision algorithm; A^{Pin} is used to represent the current pinpointing state of the pinpointing algorithm; A_0 is used to represent the initial

state of the algorithm, and B is used to represent the termination state of the algorithm. For ease of understanding, the state here can also be simply regarded as the logical implication set. The concept of V-projection is extended to A^{Pin}, and then $A_V := \{\alpha \mid \alpha{:}\varphi_\alpha \in A^{\text{Pin}}, V \vDash \varphi\}$.

Lemma 1. Let A^{Pin} and B^{Pin} be pinpointing states and V be a set of assignments. If $A^{\text{Pin}} \rightarrow^* B^{\text{Pin}}$, then $A_V \rightarrow^* B_V$.

Proof: if the reasoning rule Elim can be applied in the current state, then the B state can be obtained by deleting a duplicate clause in the clause set in the A state, and A^{Pin} and B^{Pin} contain the same logical implications only except that the labels of these implications are different, namely $A_V = B_V$. If another rule R is applied, then B is obtained by the new logical implications in A after the application of rule R, namely $A_V \rightarrow_R B_V$. Therefore, if the rule R can be applied in the pinpointing state A^{Pin}, namely $A^{\text{Pin}} \rightarrow_R B^{\text{Pin}}$, then $A_V \rightarrow_R B_V$ or $A_V = B_V$. Lemma 1 is proved. □

Since all clause labels are monotonic Boolean formulas, and $V_\top = \mathit{lab}(O)$ ensures that all assignments that make each propositional variable true meet all labels, so all pinpointing states A^{Pin} meet $A_{V_\top} = A$. Therefore, Lemma 1 means that the pinpointing algorithm does not construct new logical implications but just adds Boolean formula markers to those implications. When the consequence-based method terminates, the consequence-based pinpointing algorithm also terminates.

Lemma 2. Let A^{Pin} be the pinpointing state, and V be a set of assignments. If A^{Pin} is saturated, or no pinpointing rule can be applied to A^{Pin}, then A_V is also saturated, and no reasoning rule can be applied to A_V.

Proof: if the Elim rule can be applied in the current state, or in other words, A_V contains duplicate clauses α and β, where $A_V \subseteq A$, then there must be two such pinpointing clauses $\alpha{:}\varphi_\alpha$ and $\beta{:}\varphi_\beta$, in A^{Pin} and the Elim pinpointing rule can be applied to A^{Pin}. If other rules can be applied, or A_V contains the conditional clause required for applying the rule R, where $A_V \subseteq A$, then there must be conditional clauses of the pinpointing rule R^{Pin} in A^{Pin}. In other words, the pinpointing rule R^{Pin} can be applied to A^{Pin}. If there is a rule R that can be applied to A_V, then the pinpointing rule R^{Pin} of this rule can also be applied to A^{Pin}. Lemma 2 is proved. □

It can be proved that the above consequence-based pinpointing algorithm is indeed an axiom pinpointing algorithm that can complete reasoning tasks. The pinpointing rules are applied to O^{Pin} and D^{Pin} until no pinpointing rule can be applied, i.e. the rule is saturated, and the pinpointing state at saturation is A^{Pin}; then the clause $\alpha{:}\varphi_\alpha \in A^{\text{Pin}}$ is queried, and φ_α is the pinpointing formula of α.

Theorem 1 (the correctness of consequence-based glass-box pinpointing algorithm)**.** Given an ontology O and a logical implication α, suppose that $O \vDash \alpha$. When the pinpointing rule applied to O^{Pin} and D^{Pin} is saturated, φ_α is the pinpointing formula of α with respect to O, and the corresponding axiom set is the MinA of $O \vDash \alpha$. Specifically, O^{Pin} is the marked pinpointing clause set transformed from the ontology O; D^{Pin} is the pinpointing structure obtained after the context structure D adds markers to all clause; and φ_α is the marker of the clause α when the algorithm terminates.

Proof: to prove that φ_α is the pinpointing formula of α with respect to O, all assignments $V \subseteq \mathit{lab}(O)$ meet $(O_v, \alpha) \in P$ if and only if $V \vDash \varphi_\alpha$. □

It is assumed that $(O_v, \alpha) \in P$, namely $O_v \vDash \alpha$. Since the consequence-based pinpointing algorithm must terminate for each query axiom, and the consequence-based method is complete,

there is a saturation state B that meets $A_0 \rightarrow^* B$, where A_0 is the initial state, $\alpha \in B$. According to the conditions, $A_0^{\text{Pin}} \rightarrow^* A^{\text{Pin}}$, where A^{Pin} is the state when the pinpointing rule is saturated. Lemma 1 shows that $A_{0_V} \rightarrow^* A_V$, and Lemma 2 shows that A_V is saturated. Therefore, according to the correctness of the consequence-based method, $A_V = B$. Since $\alpha \in A_V$, $V \vDash \varphi_\alpha$. On the contrary, it is assumed that $V \vDash \varphi_\alpha$ then $\alpha{:}\varphi_\alpha \in A^{\text{Pin}}$. According to the conditions, $A_0^{\text{Pin}} \rightarrow^* A^{\text{Pin}}$, where A^{Pin} is saturated. Lemma 1 shows that $A_{0_V} \rightarrow^* A_V$. Since $V \vDash \varphi_\alpha$, $\alpha \in A_V$; since the consequence-based method is valid, $A_{0_V} \vDash \alpha$, namely $O_v \vDash \alpha$. Therefore, φ_α is the pinpointing formula of α with respect to O, and the corresponding axiom set is the MinA of $O \vDash \alpha$. Theorem 1 is proved.

2.4 Consequence-based pinpointing algorithm instances

For example, Table 2 shows the DL clause sets obtained from the axiom sets of the ontology O_2 according to the transformation rules given in Table 1, where each clause is marked with the unique propositional variables $ax1$, $ax2$, and so on, and the serial number of the clause is represented with (1), (2), and so on. Figure 2 shows the specific process of constructing the context pinpointing structure according to the consequence-based pinpointing algorithm to compute all justifications of the axiom $O_2 \vDash A \sqsubseteq D$.

Table 2 Corresponding DL clause sets of ontology O_2 according to transformation rules

$A \sqsubseteq \exists S^{-}.B$	$\rightsquigarrow$	$ax1{:}A(x) \rightarrow S(f_0(x), x)$ $ax2{:}A(x) \rightarrow B(f_0(x))$	(1) (2)	
$C_1 \sqcap C_2 \sqsubseteq \bot$	$\rightsquigarrow$	$ax3{:}C_1(x) \wedge C_2(x) \rightarrow \bot$	(3)	
$B \sqsubseteq \exists S.C_i$	$\rightsquigarrow$	$ax4{:}B_1(x) \rightarrow S(x, f_i(x))$ $ax5{:}B_1(x) \rightarrow B_2(f_i(x))$	(4) (5)	for $1 \le i \le 2$
$C_i \sqsubseteq D$	$\rightsquigarrow$	$ax6{:}C_i(x) \rightarrow D(x)$	(6)	for $1 \le i \le 2$
$B \sqsubseteq \le 2.S$	$\rightsquigarrow$	$ax7{:}B(x) \wedge \bigwedge_{1 \le i \le 3} S(x, z_i) \rightarrow \bigvee_{1 \le j < k \le 3} z_j \approx z_k$	(7)	

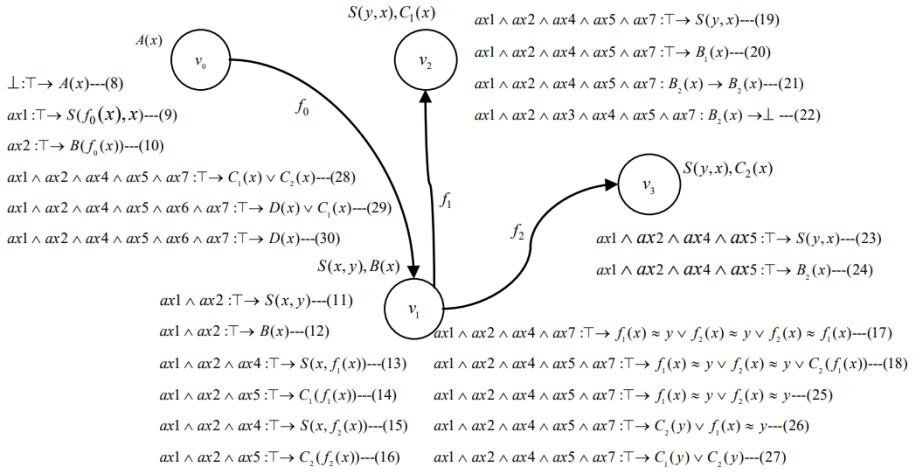

Figure 2 Specific process of judging whether $O_2 \vDash A \sqsubseteq D$ is true and computing its justifications according to the consequence-based pinpointing algorithm

First, a context v_0 is constructed and initialized to generate Clause (8), whose pinpointing formula is $\bot$, as shown in Figure 2. This ensures that each interpretation represented in the whole context pinpointing structure generated by the algorithm contains a basic term represented by A. Next, the Hyper pinpointing rule is applied to generate Clauses (9) and (10), whose pinpointing formulas are $ax1$ and $ax2$, respectively. Then, the clause $\top \rightarrow S(f_0(x), f_1(f_0(x)))$ can be

obtained from Clauses (4) and (10), but the algorithm may not easily terminate due to the increase of nesting. Therefore, the variable x in a DL clause should be mapped to the variable x in a context clause when applying the Hyper pinpointing rule so that only the logical implication of x can be obtained when the Hyper pinpointing rule is applied in each context, thus avoiding redundant derivation.

Next, the Succ pinpointing rule is used to deal with the functional symbol f_1 in Clauses (9) and (10). To determine the information to be passed to the successor, a set of successor triggers $Su(O)$ is introduced in the consequence-based pinpointing algorithm[20]. In this instance, the body of DL Clause (7) contains atoms $B(x)$ and $S(x, z_i)$, and z_i can be mapped to the predecessor or successor of x, and the context that applies the pinpointing rule to Clause (7) will be interested with the predecessor information of this clause, so $B(x)$ and $S(x, y)$ are added in $Su(O)$. The eager strategy is used when generating a new context, so the Succ rule introduces the context v_1, with $B(x)$ and $S(x, y)$ as its core. Clauses (11) and (12) are generated by the Core pinpointing rule, whose pinpointing formulas are both $ax1 \wedge ax2$. Next, the Hyper pinpointing rule is used to generate Clauses (13)–(16), whose pinpointing formulas are shown in Figure 2. At this time, there is enough information in the existing pinpointing clause set to be used to derive Clause (17) from Clauses (1), (2), (4), (7), and (8). Literals in the clause are sorted, and the rule is applied only to the maximum literal in sorting, and then Clause (18) is obtained, whose pinpointing formula is $ax1 \wedge ax2 \wedge ax4 \wedge ax5 \wedge ax7$.

Clauses (13), (14), and (18) contain the functional symbol f_2, so the context v_2 is introduced by the Succ pinpointing rule. According to Clause (14), $C_1(x)$ is true for all basic terms represented by the context v_2, so $C_1(x)$ is added to the core of v_2. On the contrary, the appearance of the atom $C_2(f_1(x))$ in the disjunction clause of Clause (18) means that it may not be true in the context v_2, so $C_2(x)$ is added to the body of Clause (21). Next, Clauses (3), (20), and (21) apply the Hyper pinpointing rule to generate Clause (22), whose pinpointing formula is $ax1 \wedge ax2 \wedge ax3 \wedge ax4 \wedge ax5 \wedge ax7$.

Clause (22) essentially expresses that $C_2(f_1(x))$ should not be true in the predecessor, and it is transmitted to the context v_1 by the Pred pinpointing rule of the form Clause (25). It is considered that Clauses (18) and (22) apply the Hyper pinpointing rule to obtain the logical implication Clause (25), and meanwhile, it is observed that the term $f_1(x)$ in the context v_1 is expressed as the variable x in the context v_2.

Clauses (14) and (26) apply the Eq pinpointing rule to generate Clause (27), which in essence expresses that the predecessor of the current context must meet $C_1(x)$ or $C_2(x)$. The set $\mathrm{Pr}(O)$[20] of the predecessor trigger contains $C_1(y)$ and $C_2(y)$, so Clause (27) can apply the Pred pinpointing rule to obtain Clause (28), whose pinpointing formula is $ax1 \wedge ax2 \wedge ax4 \wedge ax5 \wedge ax7$. Finally, the Hyper pinpointing rule can be applied twice to obtain Clause (30), whose pinpointing formula is $ax1 \wedge ax2 \wedge ax4 \wedge ax5 \wedge ax6 \wedge ax7$, and then the algorithm terminates. $O_2 \models A \sqsubseteq D$ is true, and the MinA of the logical implication is {(1), (2), (4), (5), (6), (7)}.

3 Consequence-based Black-box Pinpointing Algorithm

3.1 Solving process of the single justification of logical implication

The core of the glass-box pinpointing algorithm is to design a new pinpointing rule that can compute the label corresponding to the axiom based on the reasoning rules of the consequence-based method. Its advantage is that when the applied pinpointing rule is saturated, the pinpointing formula of the implication can be obtained and used to directly solve all justifications of the implication, without the need to repeatedly call the reasoner to judge whether the implication is true or to solve all justifications by other methods such as Hitting Set Tree (HST) after one justification is solved. However, the glass-box method requires designing the corresponding

pinpointing rule for different reasoning rules and even redesigning a new pinpointing rule as the ontology becomes more expressive. The black-box pinpointing algorithm, in contrast, can directly call the consequence-based method as the reasoner since it is not limited to specific DLs and reasoning mechanisms, and thus there is no need to design the corresponding pinpointing rule. Moreover, the black-box method can also deal with the problem of axiom pinpointing for more expressive DL ontologies without the need to consider the change of the pinpointing rule caused by new constructors. Therefore, this study proposes the consequence-based black-box pinpointing algorithm to solve the justifications of logical implications.

A MinA of a given logical implication α can be computed mainly in the expansion and contraction phases[5]. In the expansion phase, axioms are randomly selected from ontology O, and every time an axiom is selected, the consequence-based method is called to judge whether a newly constructed ontology O' can derive α. α is added to O' until $O' \vDash \alpha$, and then the expansion phase is completed. In the contraction phase, the axioms of the ontology O' obtained in the expansion phase are deleted one by one, and every time an axiom is deleted, the consequence-based method is called to judge whether O' can derive α. If $O' \vDash \alpha$, the axioms are further deleted until the new axiom set obtained after the current axiom is deleted cannot derive α; then the current axiom is added into the axiom set again, and the axiom set obtained after the above axioms are deleted one by one is one justification of α.

Since the algorithm selects axioms randomly in the expansion and contraction phases, the algorithm is less efficient. Therefore, specific axioms are often selected based on dependence in the expansion phase so that the axioms selected for the algorithm are more targeted. The common black-box method establishes a dependence axiom graph according to the concepts of conceptual dependence and axiomatic direct and indirect dependence and proposes the search strategy of selection function based on depth-first or width-first. When the solving process of the black-box pinpointing method calls the reasoner based on the consequence-based decision rule to judge whether a given logical implication can be obtained, differently from the Tableau-based black-box pinpointing method, a series of corresponding decision rules should be applied to the clause set of the ontology to judge whether the implication is true[20]. The black-box pinpointing method does not depend on a specific reasoner, and the strategy for optimization of the selection function in the common black-box pinpointing method applies to the optimization of the consequence-based black-box pinpointing algorithm in this study. Every time the reasoner is called based on the consequence-based decision rule, it is necessary to first construct a root context with Γ_Q as its core and then establish a directed graph according to the specific process in Algorithm 1 to judge whether $\Gamma_Q \rightarrow \Delta_Q$ is true. Therefore, the dependence axiom graph of Γ_Q by selection function is constructed, and the axioms in the expansion phase by the search strategy based on width-first are determined.

For the ontology O_2 in Section 2.4, for example, justifications of the axiom $O_2 \vDash A \sqsubseteq D$ are explored according to the above solving process. There are five axioms in O_2, as shown in the left-hand column of Table 2, and these axioms are marked with α_1, α_2, α_3, α_4 , α_5 from top to bottom. The dependence axiom graph of concept A is constructed. First, the axiom α_1 is selected according to the width-first strategy and added to the empty axiom set O', and the reasoning algorithm in Ref. [20] is called to judge that the logical implication $O' \vDash A \sqsubseteq D$ is not true. The axiom α_3 is further selected and added to O', and the reasoning algorithm is called to judge that the implication is still not true. The axiom α_5 is further selected according to the width-first strategy and added to O', and the reasoning algorithm is called to judge that the implication is again not true. Axioms α_2 and α_4 are successively added to O' until the implication is true in $O' = \{\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\}$, and then the expansion phase is completed. To ensure that the axiom set obtained is the MinA of the given logical implication, the contraction

phase is started. Axioms in O' are deleted one by one. First, axiom α_1 is selected from the axiom set O' and deleted to obtain the new O' and the reasoner is called to judge that the $O' \vDash A \sqsubseteq D$ is not true, and then the deleted axiom α_1 is added to the axiom set again. Then, axiom α_2 is selected from the axiom set and deleted to obtain the new O' and the reasoner is called to judge that the target axiom is still not true, and then the deleted axiom α_2 is added to the axiom set again. The above operation is repeated. When axiom α_3 is selected, the axiom set obtained is $O' = \{\alpha_1, \alpha_2, \alpha_4, \alpha_5\}$, and the reasoner is called to find that $O' \vDash A \sqsubseteq D$ is true. The above operation is repeated to search for such new O' until the search is completed. Finally, the axiom set obtained is $\{\alpha_1, \alpha_2, \alpha_4, \alpha_5\}$, which is the justification of $O_2 \vDash A \sqsubseteq D$.

Different from the pinpointing rule in Section 2.2, the decision rule only adds or deletes the required clause in the clause set without tracing the reasoning process. Different from the black-box method where lightweight DL ontologies are coded by calling a SAT solver, this solving process also requires the coding of DL ontologies, but it codes the DL expression into a multi-type clause equality logic expression to deal with more expressive ontologies. In addition to deal with expressive DL ontologies, it also improves the solving efficiency to some extent due to the superiority of the consequence-based decision reasoner.

3.2 Solving process of all justifications of logical implication

The black-box algorithm that computes all justifications of a given logical implication uses Reiter's HST[21] to expand the single justification obtained in the above justification solving process, thereby obtaining other justifications of this implication. First, the hitting set and HST are defined as follows.

Definition 1 (Hitting set[21])**.** Let F be a set cluster the hitting set of F is a set $H \subseteq \cup_{S \in F} S$, which ensures that for all $S \in F$, $H \cap S \neq \varnothing$. One hitting set of F is minimal if and only if any of its proper subsets is not the hitting set of F.

The Hitting Set Tree (HS-Tree)[21] method defines a hitting set tree T_r of a set cluster F, and both nodes and edges of the tree are marked, of which the node is marked by an element in F, and the edge is marked by an element in the marker of its predecessor node. With an element in F as the root node of the tree, nodes in the tree are processed successively, and then the node is marked, and new nodes are constructed to build the HS-Tree.

$P(n)$ is defined as the set of edge markers on the path from the root node to the node n. If there is an element in the set cluster F whose intersection with $P(n)$ is an empty set, then this element is used as the marker of the node n; otherwise, the node n is marked as "$\surd$". If n is a node marked as "$\surd$", then $P(n)$ is a hitting set of the set cluster F, and each minimal hitting set of F is the $P(n)$ of the node n marked as "$\surd$" in its corresponding hitting set tree. The $P(n)$ corresponding to the node n marked as "$\surd$" does not contain all hitting sets of F, but it contains all minimal hitting sets of F.

F gives all justifications of a given logical implication, and the solving process of a single justification needs to be called to compute one justification of F in access to F. To produce an HST that is as small as possible, only the minimal hitting set of F is given, and the HST with the least F is accessed. The solving process of all justifications includes two methods to reduce the number of calls to the solving process of a single justification, namely early path termination and repeated interpretation.

4 Experimental Analysis

4.1 Experimental data

The experiment in this section aims to verify the feasibility and effectiveness of the glass-box and black-box pinpointing algorithms proposed in this study. Reasoning tools CBPin_Glass_Box

and CBPin_Black_Box are designed according to the two methods and compared with the other reasoning tools Pellet[22], FaCT++[23], and EL2SAT[24] to analyze the merits and demerits of the system in DL ontologies with different expressive powers. The experiment is conducted on a PC VMware Workstation 64-bit Fedora system (2 GB of RAM), and the data used for the experiment are from a standard ontology base, mainly the Open Biological Ontology (OBO) Foundry, the Gardiner Ontology Suite, the Phenoscape Project, and variants of the GALEN Ontology. All ontologies are preprocessed to analyze the ontology import so that all test ontologies can be included in a single file and loaded through OWL API. Since the experiment is conducted based on consistent ontologies to compute the justifications of a given logical implication, no inconsistent ontologies are considered nor ontologies that are too simple to provide any useful performance tests (namely, those including few axioms or classes). And since the proposed pinpointing algorithms are mainly for expressive DL ontologies, ten ontologies shown in Table 3 are selected from http://www.cs.ox.ac.uk/isg/ontologies, and their ID, name, DL expressive power, number of classes, number of attributes, and number of axioms in TBox are shown. These ten ontologies include strongly expressive ones, such as ontology 00511, and weakly expressive ones, such as ontology 00365 so that the efficiency of axiom pinpointing of reasoning tools proposed in this study can be compared in ontologies with different expressive powers.

Table 3 Experimental data set

ID	Name	Expressive power	Number of classes	Number of attributes	Number of Axioms in TBox
00004	BAMS-simplified	SHIF	1,110	12	18,813
00011	Biopax-level2	ALCHN(D)	41	33	333
00013	CommonSenseMapping	SHIN(D)	126	273	981
00015	DOLCE-Lite	SHI	37	70	279
00031	Galen-ians-full-	EL++	2,749	413	4,205
00040	GO_extensions-anatomy	SRIQ	58,882	220	130,376
00365	OBO-bfo	ALC	39	0	95
00511	OBO-kisao	ALCHIQ(D)	225	9	698
00512	OBO-lipid	ALCHIN	716	46	2,349
00582	OBO-poro	SRQ	642	17	798

The ten ontologies shown in Table 3 are of different expressive powers, of which ontology 00031 and ontology 00365 are the least expressive, and their axioms can be pinpointed with these reasoning tools, while the remaining eight expressive ontologies cannot be dealt with by the EL2SAT reasoner. The EL2SAT reasoner is efficient for lightweight DL ontologies, but it applies only to weakly expressive DL ontologies. Therefore, it is not analyzed in the following experiment.

4.2 Experimental results and analysis

Fifty query axioms are randomly selected from the ten ontologies shown in Table 3, and CBPin_Glass_Box, CBPin_Black_Box, Pellet, and FaCT++ are used to compute all justifications of these axioms, and the average solving time is recorded. Figure 3 shows the average time of the CBPin_Glass_Box and the CBPin_Black_Box algorithms in computing all justifications of axioms randomly selected from the ten ontologies in Table 3.

According to Figure 3, the glass-box pinpointing algorithm is more efficient than the black-box pinpointing algorithm, which is because the former modifies one execution process of the decision algorithm and uses the pinpointing rule to establish the correspondence between the label of the Boolean formula of the clause and all MinAs of the logical implication, while the latter needs to repeatedly call the consequence-based method, which is time-consuming

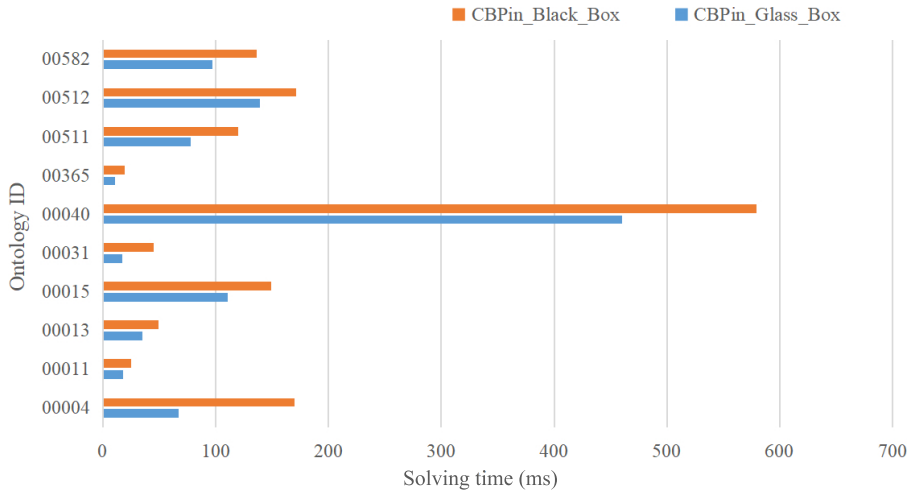

Figure 3 Time comparison between two consequence-based axiom pinpointing algorithms in computing all justifications of different ontologies

and inefficient. Then, the glass-box pinpointing algorithm CBPin_Glass_Box with better performance is compared with the existing Pellet and FaCT++ reasoners, as shown in Figure 4.

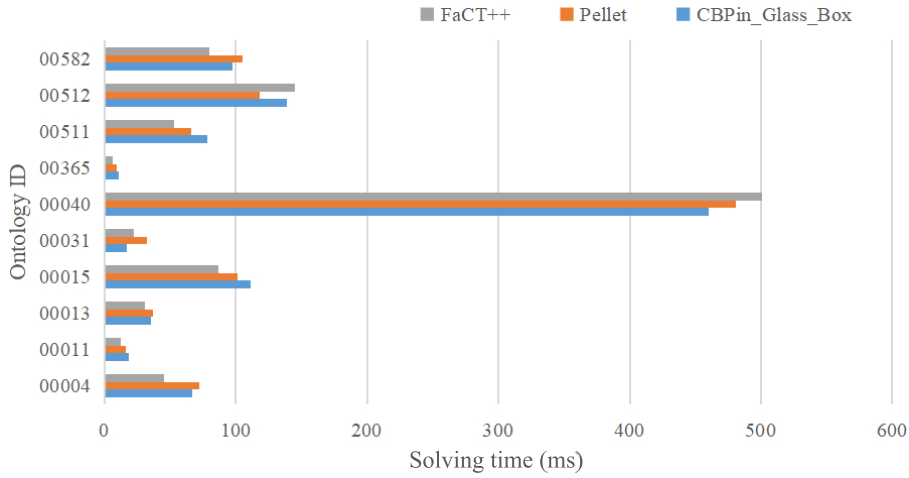

Figure 4 Time comparison between consequence-based pinpointing algorithms and other reasoning tools in computing all justifications of different ontologies

The consequence-based glass-box pinpointing algorithm is superior; different from the Tableau-based pinpointing method which needs to trace clashes by interpretation and then backtrack the axioms, it directly traces the specific process of reasoning by pinpointing rule so that all MinAs of the given logical implication can be obtained while judging whether this implication is true. However, Pellet and FaCT++ are solving tools highly optimized by researchers, and the glass-box pinpointing tool proposed in this study only basically realizes the consequence-based glass-box pinpointing algorithm rather than optimizing the efficiency. Therefore, according to the experimental results in Figure 4, although Pellet and FaCT++ reasoners solve the axioms faster than the CBPin_Glass_Box algorithm, the latter also shows its merits in ontology 00040. Similarly, the glass-box algorithm solves the axioms faster than

Pellet in ontologies 00004, 00013, 00031, and 00582 although it is less efficient than FaCT++. Therefore, it can be seen that the consequence-based glass-box pinpointing algorithm is effective to some extent and thus is worth studying.

5 Summary

This study presented the consequence-based glass-box and black-box axiom pinpointing methods for expressive DL ontologies. The glass-box method uses the modified consequence-based decision rule to obtain the axiom pinpointing rule and introduces the concept of pinpointing formula to establish the correspondence between the label of the Boolean formula of the clause and all the MinAs of the logical implication. At the same time, the correctness of this method is proved. Since the glass-box pinpointing algorithm needs to be redesigned for the constructor when the expressive power of DL ontologies continues to increase, this study further presented the black-box axiom pinpointing method that directly calls the consequence-based reasoner, which completes axiom pinpointing by computing a single justification and all justifications. This study also designed reasoning tools based on black-box and glass-box pinpointing algorithms and conducted experiments on DL ontologies with different expressive powers to verify that consequence-based pinpointing algorithms have advantages on expressive DL ontologies.

References

[1] Baader F, Calvanese D, McGuinness D, Nardiand D, Patel-Schncider PF. Basic Description Logics. 2nd ed., Cambridge: Cambridge University Press, 2007. 47–100.

[2] Suntisrivaraporn B, Baader F, Schulz S, Spackman KA. Replacing SEP-triplets in SNOMED CT Using Tractable Description Logic Operators. Proc. of the 11th Conf. on Artificial Intelligence in Medicine. Amsterdam: Springer, 2007. 287–291.

[3] Schlobach S, Cornet R. Non-standard reasoning services for the debugging of description logic terminologies. Proc. of the 18th Int'l Joint Conf. on Artificial Intelligence. 2003. 355–362.

[4] Kalyanpur A, Parsia B, Sirin E, Hendler JA. Debugging unsatisfiable classes in OWL ontologies. Journal of Web Semantics, 2005, 3(4): 268–293.

[5] Kalyanpur A, Parsia B, Horridge M, Sirin E. Finding all justifications of OWL DL entailments. Proc. of the 16th Int'l Semantic Web Conf. Heidelberg: Springer-Verlag, 2007. 267–280.

[6] Ouyang DT, Su J, Ye YX, Cui XJ. The ontology debugging method based on concept R-MUPS. Ruan Jian Xue Bao/Journal of Software, 2015, 26(9): 2231–2249 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4735.htm [doi: 10.13328/j.cnki.jos.004735]

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

[8] Ji Q, Qi GL, Haase P. A relevance-directed algorithm for finding justifications of DL entailments. Proc. of the 4th Asian Conf. on the Semantic Web. Shanghai: Springer, 2009. 306–320.

[9] Zhang Y, Ouyang DT, Ye YX. Debugging and repairing incoherent ontologies based on the clash path. Ruan Jian Xue Bao/Journal of Software, 2018, 29(10): 2948–2965 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5550.htm [doi: 10.13328/j.cnki.jos.005550]

[10] Zhang Y, Ouyang DT, Cui XJ, Ye YX. Semi-models based justifications detection for OWL ontologies. Chinese Journal of Computers, 2018, 41(12): 2710–1733 (in Chinese with English abstract).

[11] Gao J, Ouyang DT, Ye YX. Exploring duality on ontology debugging. Applied Intelligence, 2020, 50(2): 620–633.

[12] Baader F, Peñaloza R, Suntisrivaraporn B. Pinpointing in the description logic EL. Proc. of the 2007 Int'l Workshop on Description Logics. Brixen-Bressanone, 2007. 52–67.

[13] Gao MY, Ye YX, Ouyang DT, Wang B. Finding justifications by approximating core for large-scale ontologies. Proc. of the 28th Int'l Joint Conf. on Artificial Intelligence (IJCAI-19). 2019. 6432–6433.

[14] Ye YX, Cui XJ, Ouyang DT. Exracting a justification for OWL ontologies by critical axioms. Frontiers of Computer Science, 2020, 14(4): 144305. [doi: 10.1007/s11704-019-7267-5]

[15] Baader F, Brandt S, Lutz C. Pushing the EL envelope. Proc. of the 19th Int'l Joint Conf. on Artificial Intelligence. Edinburgh, 2005. 364–369.

[16] Simančík F, Kazakov Y, Horrocks I. Consequence-based reasoning beyond Horn ontologies. Proc. of the 22nd Int'l Joint Conf. on Artificial Intelligence. Barcelona, 2011. 1093–1098.

[17] Simančík F, Motik B, Horrocks I. Consequence-based and fixed-parameter tractable reasoning in description logics. Artificial Intelligence, 2014, 209: 29–77.

[18] Cucala DT, Grau BC, Horrocks I. Pay-as-you-go consequence-based reasoning for the description logic SROIQ. Artificial Intelligence, 2021, 298: 103518.

[19] Ozaki A, Peñaloza R. Consequence-based axiom pinpointing. Proc. of the 12th Int'l Conf. on Scalable Uncertainty Management. Milan, 2018. 181–195.

[20] Bate A, Motik B, Grau BC, Simančík F, Horrocks I. Extending consequence-based reasoning to SRIQ. Proc. of the 15th Int'l Conf. on the Principles of Knowledge Representation and Reasoning. Cape Town, 2016. 187–196.

[21] Reiter R. A Theory of Diagnosis from First Principles. Artificial Intelligence, 1987, 32(1): 57–95.

[22] Sirin E, Parsia B, Grau B C, Kalyanpur A, Katz Y. Pellet: A practical OWL-DL reasoner. Journal of Web Semantics, 2007, 5(2): 51–53.

[23] Tsarkov D, Horrocks I. FaCT++ description logic reasoner: System description. Proc. of the 3rd Int'l Joint Conf. on Automated Reasoning. Seattle: Springer-Verlag, 2006. 292–297.

[24] Sebastiani R, Vescovi M. Axiom pinpointing in large EL+ ontologies via SAT and SMT techniques. Technical Report, DISI-15-010, University of Trento, 2015.

Jing Li, master's degree candidate. Her research interests include semantic Web and automated reasoning.

Yuxin Ye, Ph.D., professor, doctoral supervisor. His research interests include semantic Web and automated reasoning.

Dantong Ouyang, Ph.D., professor, doctoral supervisor. Her research interests include model diagnostic checking and automated reasoning.